FINGER CNC

Open Designs Customize The Future

# HMI Designer Teaching Manual

Version No: F202407HT-EN

Guangzhou Finger Technology Co., Ltd

## CONTENTS

# 1. Introduction

## 1.1. Revision History

| Modification Date | Note |
|---|---|
| 20240507 | Initial Draft |
| | |
| | |
| | |

## 1.2. About FINGER CNC

Guangzhou Finger Technology Co., Ltd. aims to develop high-performance open CNC systems that simplify automation development and make machine tool automation easily accessible. As one of China's high-performance controller manufacturers, Finger Technology focuses on customer needs, continuously pushing the boundaries of technological R&D to gradually form a comprehensive ecosystem of key automation technologies, providing customers with complete solutions and convenient services. We have established complete, professional, and efficient sales and service channels across various regions in China.

Finger Technology is dedicated to the R&D and production of CNC systems, motion controllers, edge computing controllers, Open CNC development platforms, CAD/CAM technologies, machine vision technologies, robotic control technologies, and industrial IoT technologies. Our industry-leading Open CNC development platform simplifies the customization of machinery and creates unique product value for our customers. Finger Technology proposes the integration of six core embedded technologies (motion control, HMI, PLC, machine vision, CAD/CAM, IoT) as part of our integrated product solutions, providing customers with optimal automation solutions. We have accumulated extensive

product experience and customer base in industries such as turning and milling centers, grinding machines, spring machines, tool machines, woodworking machinery, winding machines, spinning machines, pipe bending machines, and 3C electronics, continuously striving for excellence.

In the field of high-speed and high-precision, Finger Technology conducts in-depth research on various high-performance motion control algorithms, widely applicable to different industry needs. Especially in multi-axis linkage interpolation, RTCP five-axis linkage control, multi-axis multi-channel control, electronic cam, winding, and tension control technologies, we provide customers with a wider range of solutions.

Focusing on customer needs, emphasizing results, pursuing excellence in innovation, and respecting talent have been the core principles and values of Finger Technology since its inception. We remain dedicated to these principles, diligently advancing and consistently developing reliable, user-friendly automation products. By extending the Open CNC concept to customer endpoints, we create exclusive value for our clients.

# 2. B-Series HMI Software Overview

## 2.1. Software Overview

FINGER HMI Designer is the next-generation human-machine interface (HMI) development tool for the B-Series automatic control systems.

## 2.2. Interface Introduction

The main interface of the FINGER HMI Designer software is shown in Figure 2.2-1.

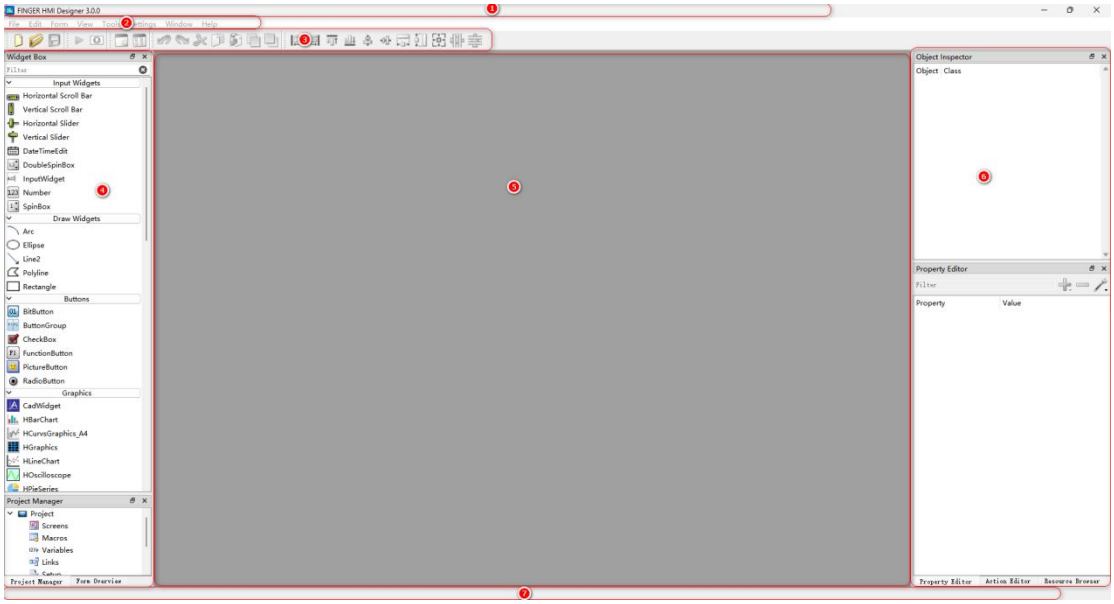


Figure 2.2-1 Software Main Interface

Here is the translation of the description of each area:

1. Title Bar: Displays software version, file path, and other information.
2. Menu Bar: Provides access to most software functions, including File, Edit, Screen, View, Tools, Settings, Window, and Help buttons.
3. Toolbar: Offers various editing tools for user convenience and enhanced window design efficiency.
4. Auxiliary Windows:

1) Users can select which auxiliary windows to display under Menu Bar -> View. They can also drag the window title bar to move the auxiliary window above or below other auxiliary windows. These include:
2) Screen Component Box: Contains all plugins; users can drag and drop plugins into the user editing area.
3) Project Management: Allows viewing of screen, macro, and variable settings.
4) Screen Overview: Provides a preview of all project screen contents.

5. User Area: The main editing area where users can add various plugins to windows, which serve as the basic containers.
6. Auxiliary Windows:

7. Object Viewer:

1) Displays all plugin classes and object names added to the current screen. Clicking on an object name allows quick navigation to the specific plugin.

2) Action Editor: Displays events and macros associated with corresponding plugins.

3) Resource Browser: Allows saved plugins, events, and macros to be added to other projects.

8. Status Bar: Displays temporary information.

## 2.3. Installation Overview

# 3. Basic Operations

## 3.1. Files

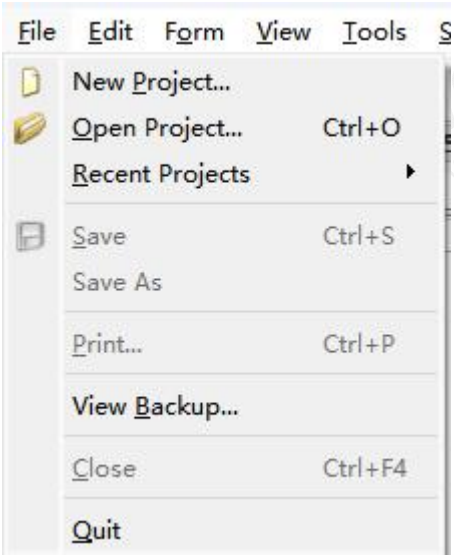



Figure 3.1-1 File Function Menu

### 3.1.1. New Project

### 3.1.2. Open Project

### 3.1.3. Recently Opened

### 3.1.4. Save and Save As

### 3.1.5. Print

### 3.1.6. View Backup

1. The software will automatically backup the project files that are being edited by the user. If the user needs to use a backup file, they can do so through the "View Backup" option under the "File" menu.
2. Clicking on this option will open a backup file dialog box. Users can browse through the files that have been automatically backed up by the software. Clicking on the desired file and then clicking "Open" will prompt a dialog box asking the user whether to save the currently edited file. Choosing "Save" or "Discard" will close the currently edited file and open the selected backup file for editing.

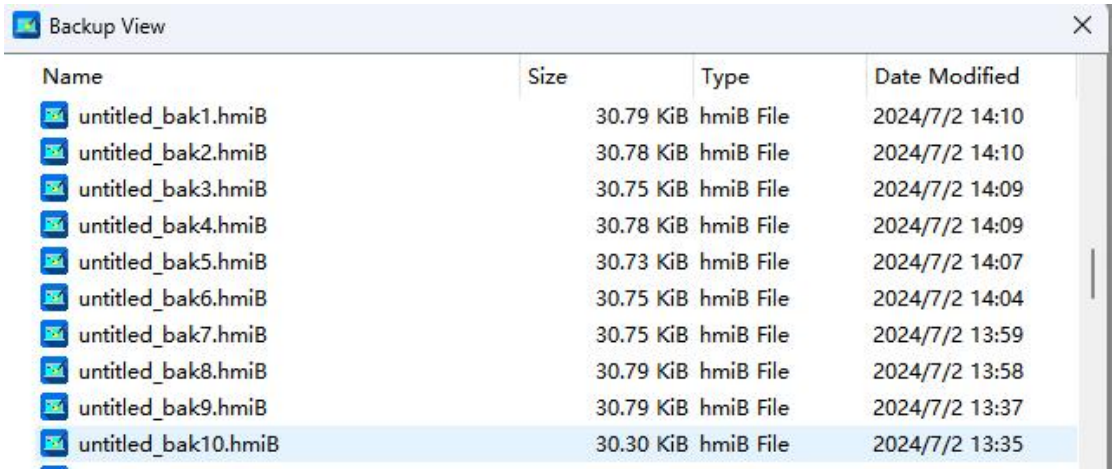



Figure 3.1.6-1 Backup Viewing

## 3.2. Edit

## 3.3. Screen

### 3.3.1. New Screen

When users need to create a new screen within the current project, they can use this function. Clicking on the New option under the Screen menu will prompt a new empty screen to appear under the current project. The default name for this screen is "Form," and users can configure its properties in the attribute settings box.

### 3.3.2. View Screen

To view screens within the project, users can click on the View Screen option. This action will open a View Form dialog box within the software, listing all the screen names currently present in the project.

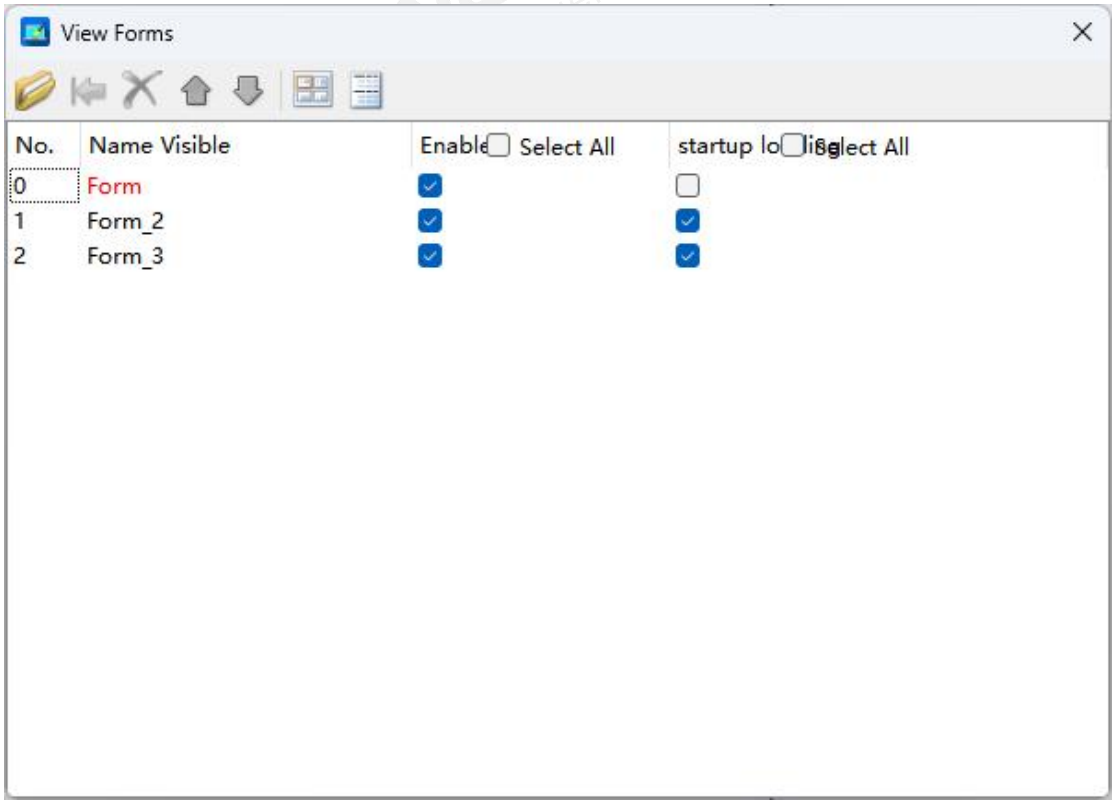


Figure 3.3.2-1: View Form Dialog Box

1. When opening the dialog box, users need to first click on the entry of the form in the list, and then proceed with the corresponding settings. Hovering the mouse over the button graphics will display the respective button's Chinese description below the pointer.
2. The list includes options for "Boot Load" and "Enable". Unchecking "Boot Load" means that when the project is uploaded to the controller, this form will not be loaded during project initialization. Compared to loading forms, this reduces project loading time, making it faster, smoother, and quicker to enter the initial form screen.
3. Users can set the form to "Enable". If "Enable" is not checked, the form cannot be used on the controller after uploading.
4. Clicking on the form name in the list and then clicking the button allows the selected form to be set as the start form, which will be displayed as the first screen after startup. The start form is marked in red in the list to distinguish it for the user.

### 3.3.3. Preview

Click on the selected form in the main window, then click on the Preview option under the Screen menu to preview the effect of this form after uploading it to the controller.

### 3.3.4. Save Image

After selecting the form, click on the Save Image option. The software will generate a PNG image that displays the content of the selected form.

### 3.3.5. Import

This feature allows you to import screens from another project into the current project. Click on this button, and a Select Project dialog box will appear. Choose the project file

and click OK. This will open an Import Objects dialog box with tabs for Screens and Macros. Select the screens and macros you want to import (hold Ctrl to select multiple items), then click Import to import the selected macros and screens into the current project file.

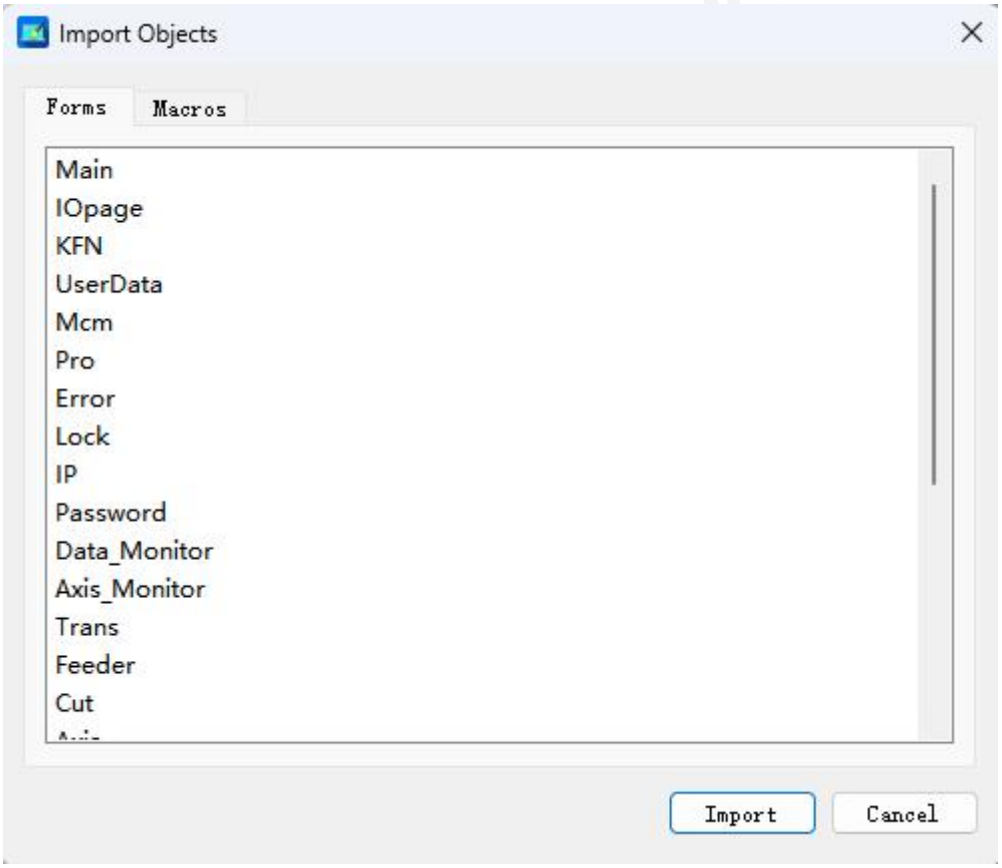



Figure 3.3.5 Select Import Content

## 3.3.6. Window Settings

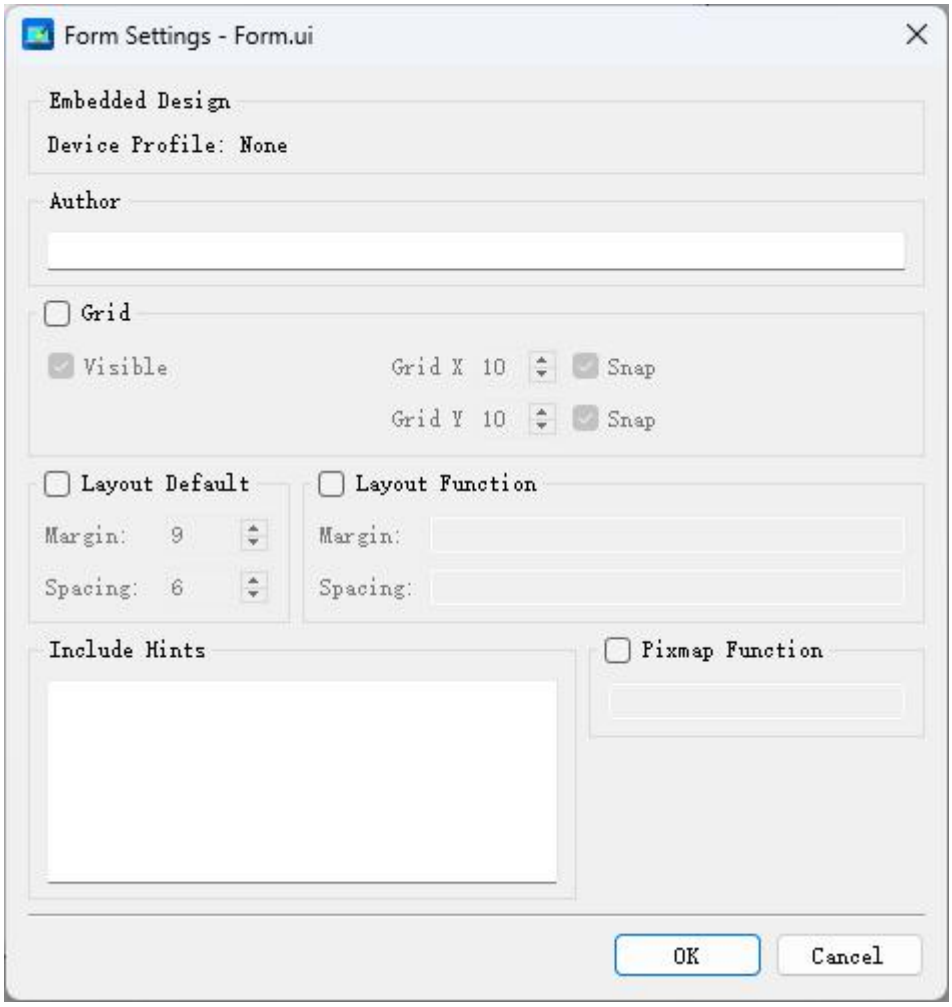



Figure 3.3.6 Window Settings

1. In the "Window Settings" dialog box, users can set options such as author name, grid, layout, and more. When the "Grid" option is unchecked, users cannot configure grid settings. The selected window will display the grid by default in the Windows software. When "Grid" is checked, users can adjust its visibility. If unchecked, the selected project window will no longer display the grid. On the right side, users can adjust the grid size, with X representing the horizontal direction and Y representing the vertical direction. After configuring these settings, the pixel spacing between points in the visible window will change accordingly.
2. In the "Window Settings" dialog, there are layout options available to users. They can

configure either the "Default Layout" or the "Layout Functionality."

## 3.4. View

## 3.5. Tools

### 3.5.1. Run Project

### 3.5.2. Download to Device

### 3.5.3. Generate Installation Package

### 3.5.4. Script Editor

### 3.5.5. MacroC Editor

### 3.5.6. Component Property Editor

### 3.5.7. View Variables

After selecting "View Variables," the software will display a "View Variables Window." Once the relevant settings are configured, the table below will show the system variables that have been used in the current project file.

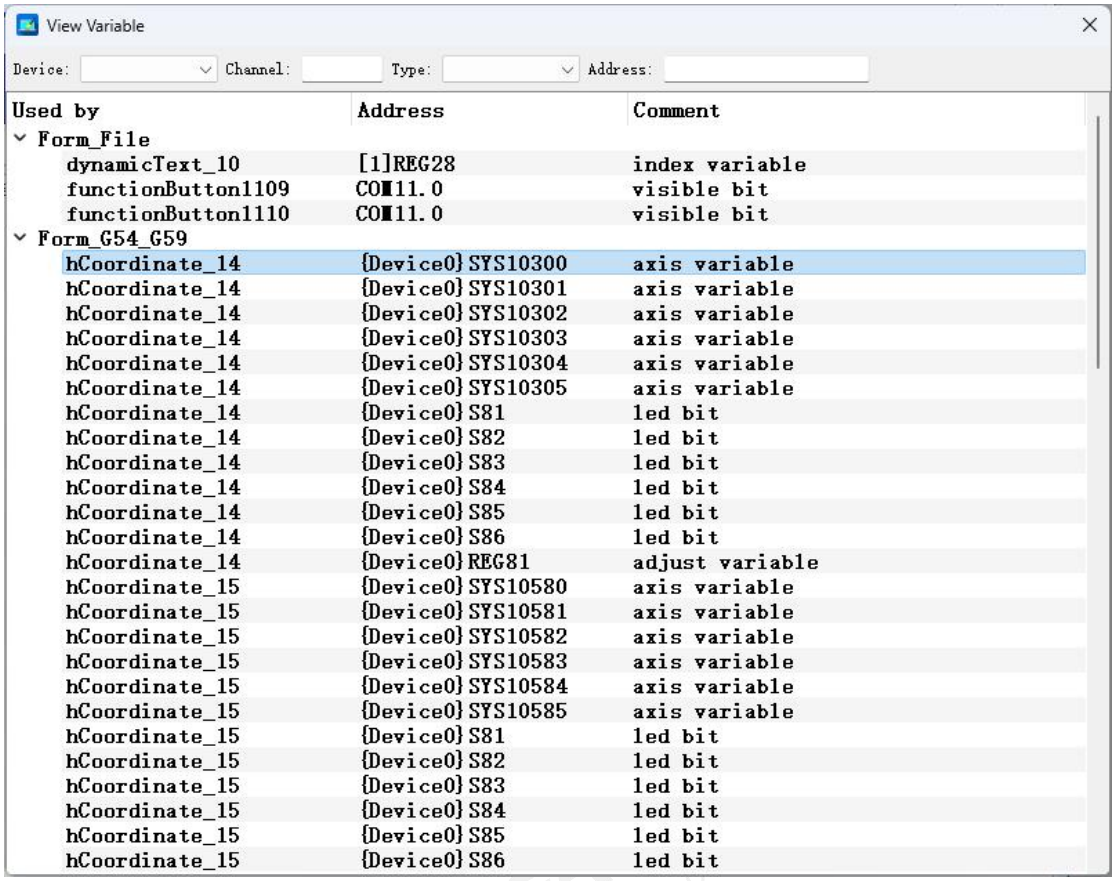



Figure 3.5.7 - View Variables Window

Users can also input channel number (range 1~16), type, and address below the title to filter. Once these three items are entered, the table will display only the system variables that meet the specified criteria.

## 3.5.8. Operation Record Settings

This feature allows users to enable operation recording for each plugin in the project screen. Once enabled, actions triggered by the plugin can be viewed in the Operation Record plugin.

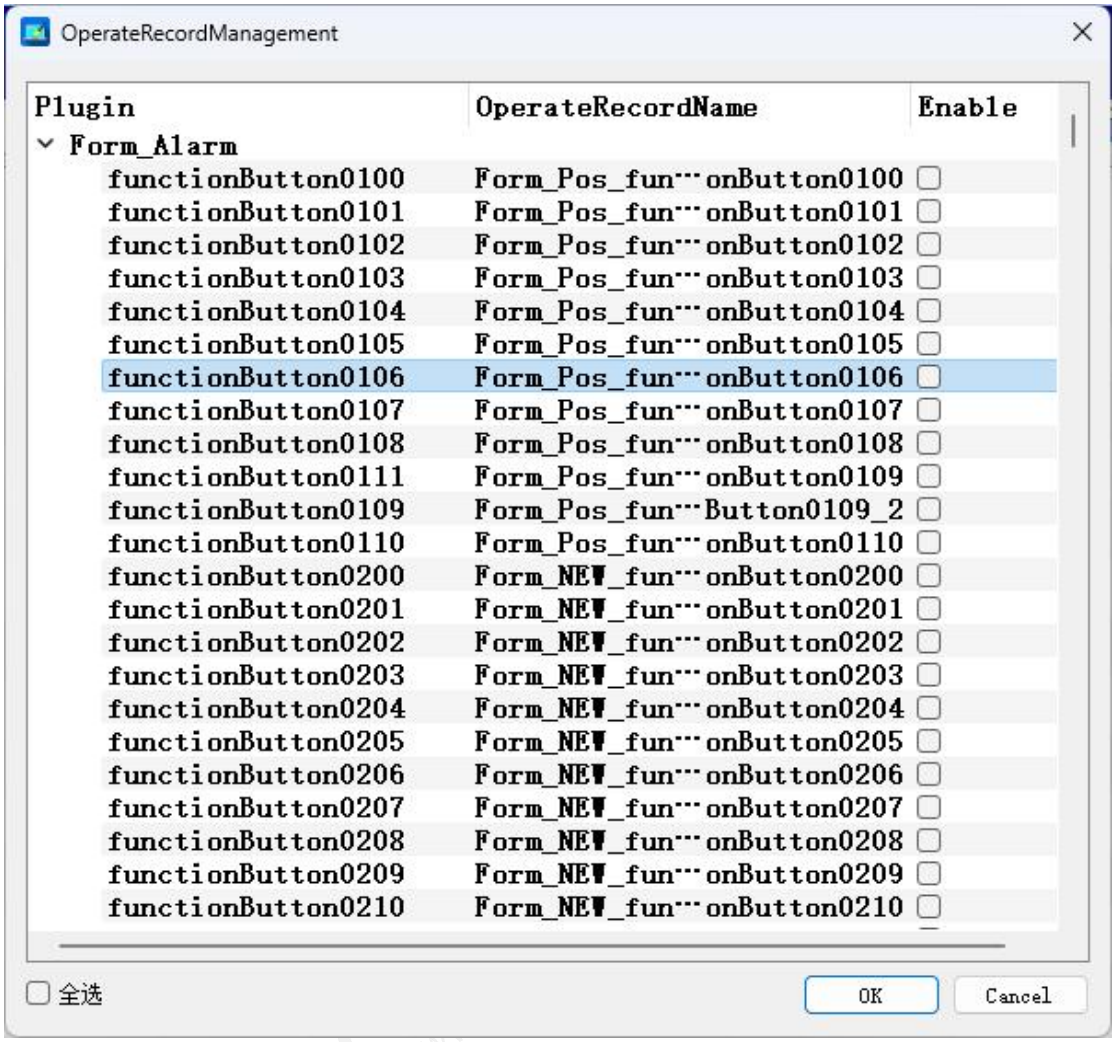



Figure 3.5.8-Operation Record Management Dialog

## 3.5.9. Clean Resources

After removing or replacing images used by plugins like image buttons or dynamic images, any unused resources can be viewed in the "Tools" menu under "Clean Resources". Resources that are currently in use will not be displayed in this list. Cleaning up unused resources can help reduce the size of the project.

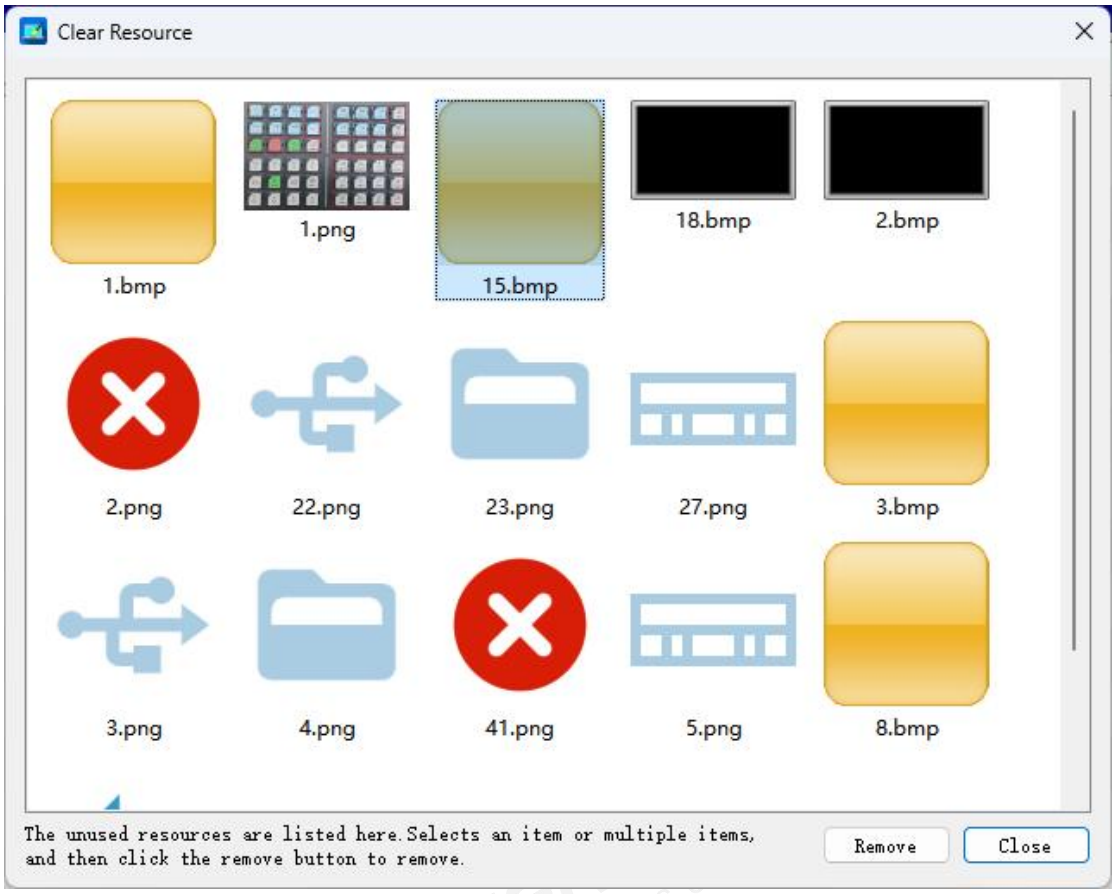
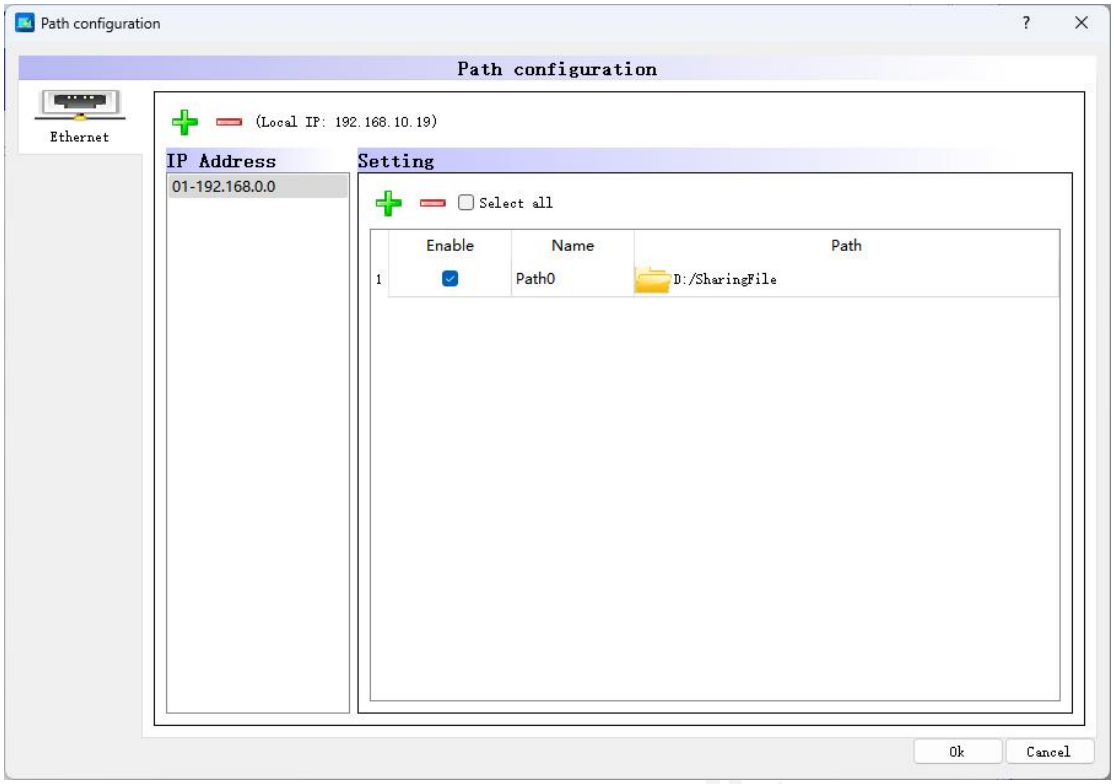



Figure 3.5.9 Clean-up Resources Dialog Box

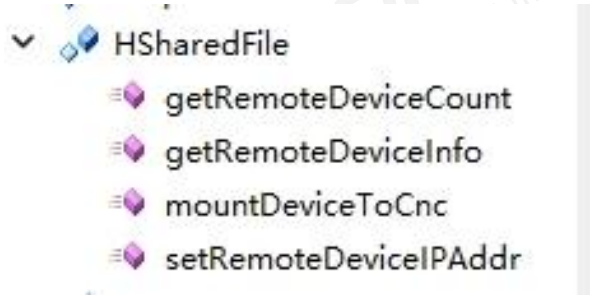
## 3.5.10. Regular Expression Editor

## 3.5.11. Shared File Settings

Shared files allow the controller to load disk files from the PC side, used in conjunction with the File Selector plugin. To use this feature, the NFS service on the PC side needs to be enabled, an NFS server needs to be set up, and the "Nfs_Server.axBin" and HMI Designer need to be updated to configure the shared file path. For specific steps, refer to the "Shared Folder Tutorial" which provides detailed configuration instructions, required system updates, NFS server software, and more.




Using the Macro Wizard in conjunction with the HSharedFile function group, you can set the computer's IP address, query the number of remote devices, retrieve remote device information, and remount folders.



### 3.5.12. Data Backup and Update

### 3.5.13. Component Native Text Translation

### 3.5.14. Export Translation

### 3.5.15. Import Translation

### 3.5.16. Terminology Library Setting

## 3.6. Setting

### 3.6.1. Software Language

### 3.6.2. Language Setting

### 3.6.3. Project Setting

Users can configure project attributes when creating a new project or modify them after project creation. Click on the Settings option in the menu bar, then select Project Settings from the dropdown menu. This will open the "Project Settings" dialog box, as shown in Figure 3.6.3-1.

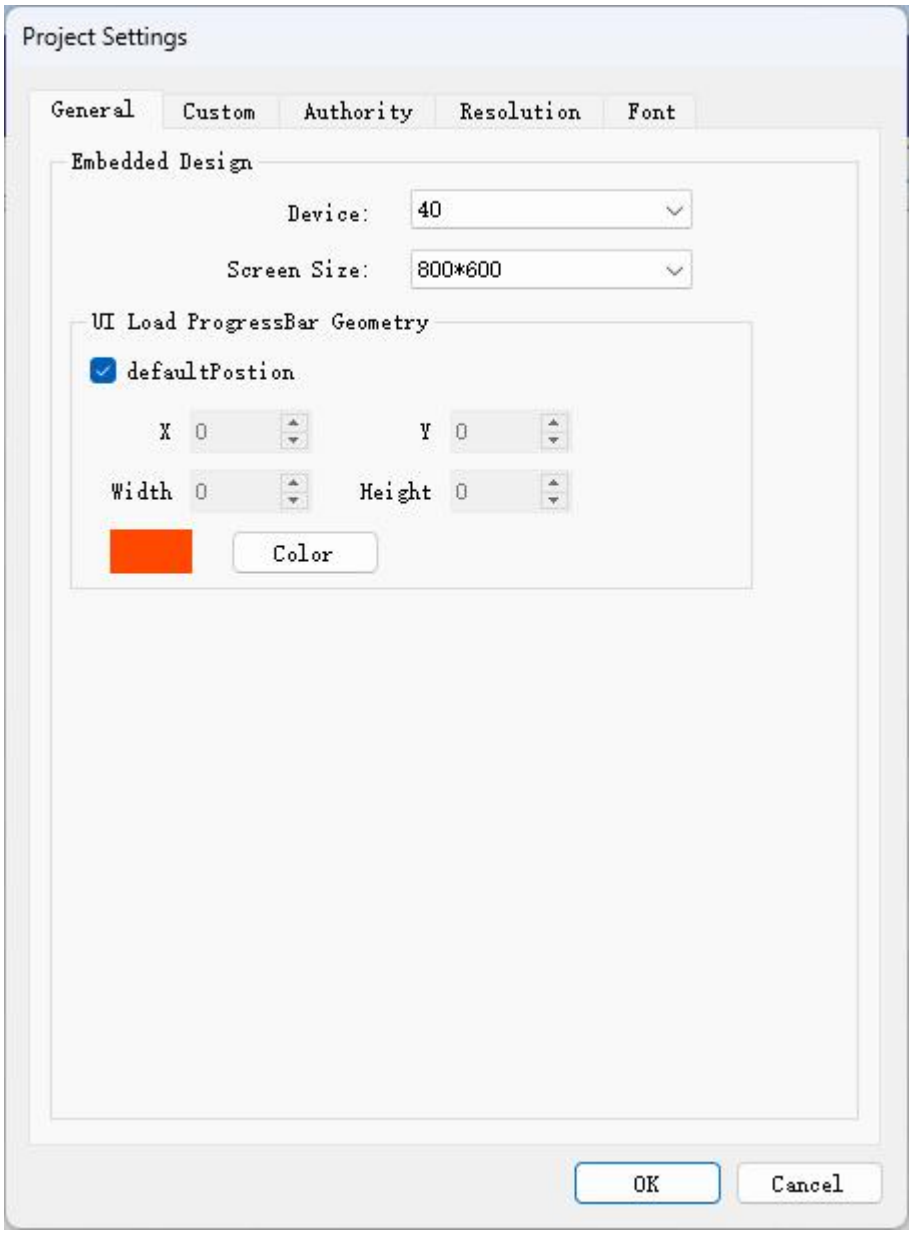



Figure 3.6.3-1 Project Settings Dialog Box

The "Project Settings" dialog box includes several tabs: "General," "Custom," "Permission Settings," "Screen Size Adjustment," and "Font." Users can adjust project properties based on their specific needs.

**[General Tab]**

**Axis Number:** Options include 40-axis and 64-axis, with 40-axis as the default setting. When creating a project, ensure to use the corresponding system data for the selected

axis configuration. Default system data is compatible with 40-axis projects, while system data with "64-axis" suffix is used for 64-axis projects. Verify the controller project axis number in the bottom right corner of the Trans software.

**Screen Size:** Users can select the screen resolution from the dropdown menu. The default is 800*600. It is recommended to adjust the screen resolution according to the actual size of the controller.

**Default Position and Size:** When enabled, the second boot progress bar is displayed by default, as shown in Figure 3.6.3-2.


Figure 3.6.3-2 Default Progress Bar Position and Size

X: Selects the X-axis (horizontal) position of the top-left corner of the second boot progress bar.

Y: Selects the Y-axis (vertical) position of the top-left corner of the second boot progress bar.

Width: Sets the width of the second boot progress bar.

Height: Sets the height of the second boot progress bar.

Color: Sets the color of the second boot progress bar, defaulting to red.

**[Customization Tab]**

**Formview Title:** Sets the title of Formview.

**Formview Icon:** Sets the icon of Formview.

**Startup Screen:** Allows changing the default startup screen image.

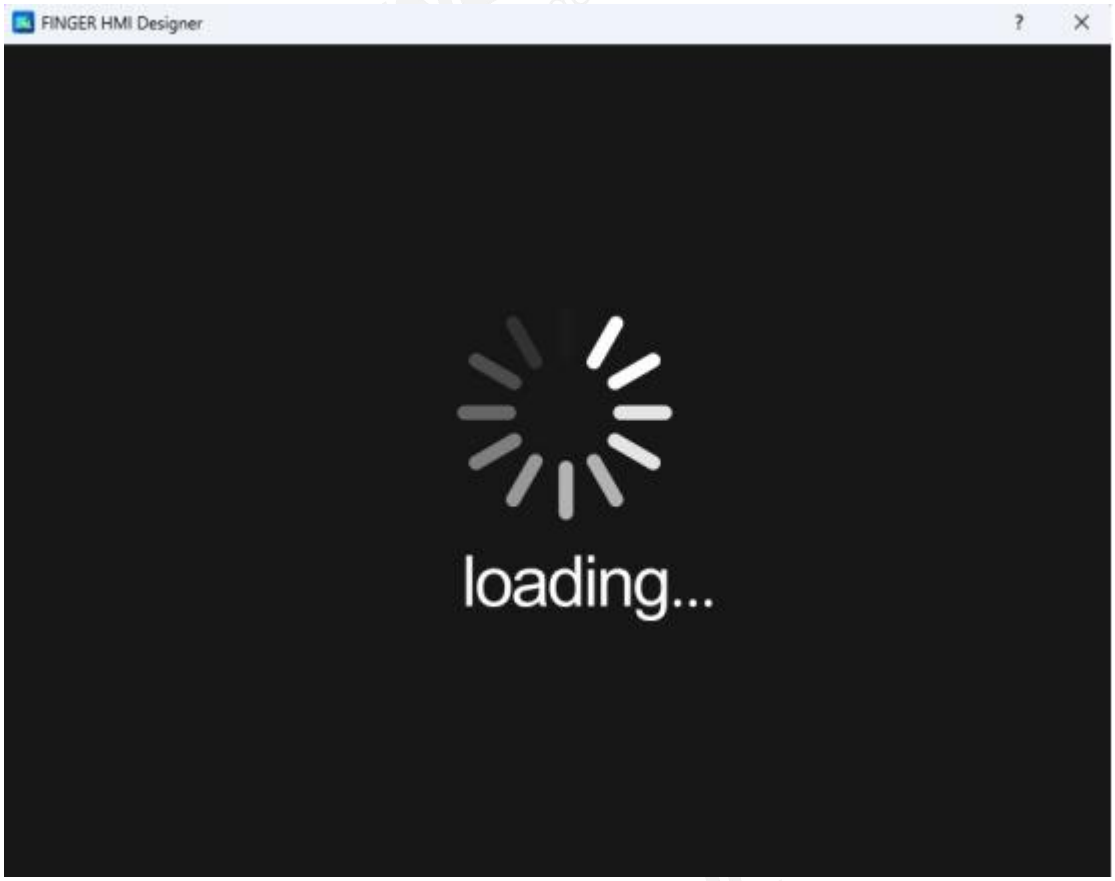



Figure 3.6.3-3 Default Startup Screen

**[Permissions Settings]**

Configure controller permissions, used in conjunction with the permission management plugin. Set levels here and manage login/switching of permissions through the plugin.

**Reset Permission Levels and Passwords:** When selected, entering the controller will remove previous settings and apply new permission levels and passwords.

**Restore Default Level on Startup:** When selected, the chosen level becomes the default

after every startup. Lower levels have higher priority; Level 0 indicates no permission settings.

**Set Number of Levels and Corresponding Passwords:** After setup, view configured levels in the permission management plugin drop-down.

**Add and Remove Levels:** First add a level column, then set level names and passwords.

**[Screen Size Adjustment]**

The screen size in the user's project needs to be adjusted according to the controller's screen size. If adjustments are necessary for the screen size set during project creation, users can modify it under the "Screen Size Adjustment" tab in the "Project Settings" dialog accessed from the "Settings" menu.

Users can directly modify the screen size of the current project or choose to do so through "Save as New Project". If "Modify Current Project" is selected and confirmed, the screen size of the current project's screens will be adjusted. If "Save as New Project" is chosen, the size of screens in the currently edited project will remain unchanged. The software will prompt a dialog to set a name and storage path. A new project file will be generated at the specified location, with screen sizes adjusted accordingly.

**[Font]**

The Font tab in the Project Settings dialog allows users to set fonts for the project. Users can set 5 types of fonts here. Double-click the table box under the Enable column, and it will turn into a dropdown box style. Users can click on the dropdown to select a font, as shown in Figure 3.6.3-4.

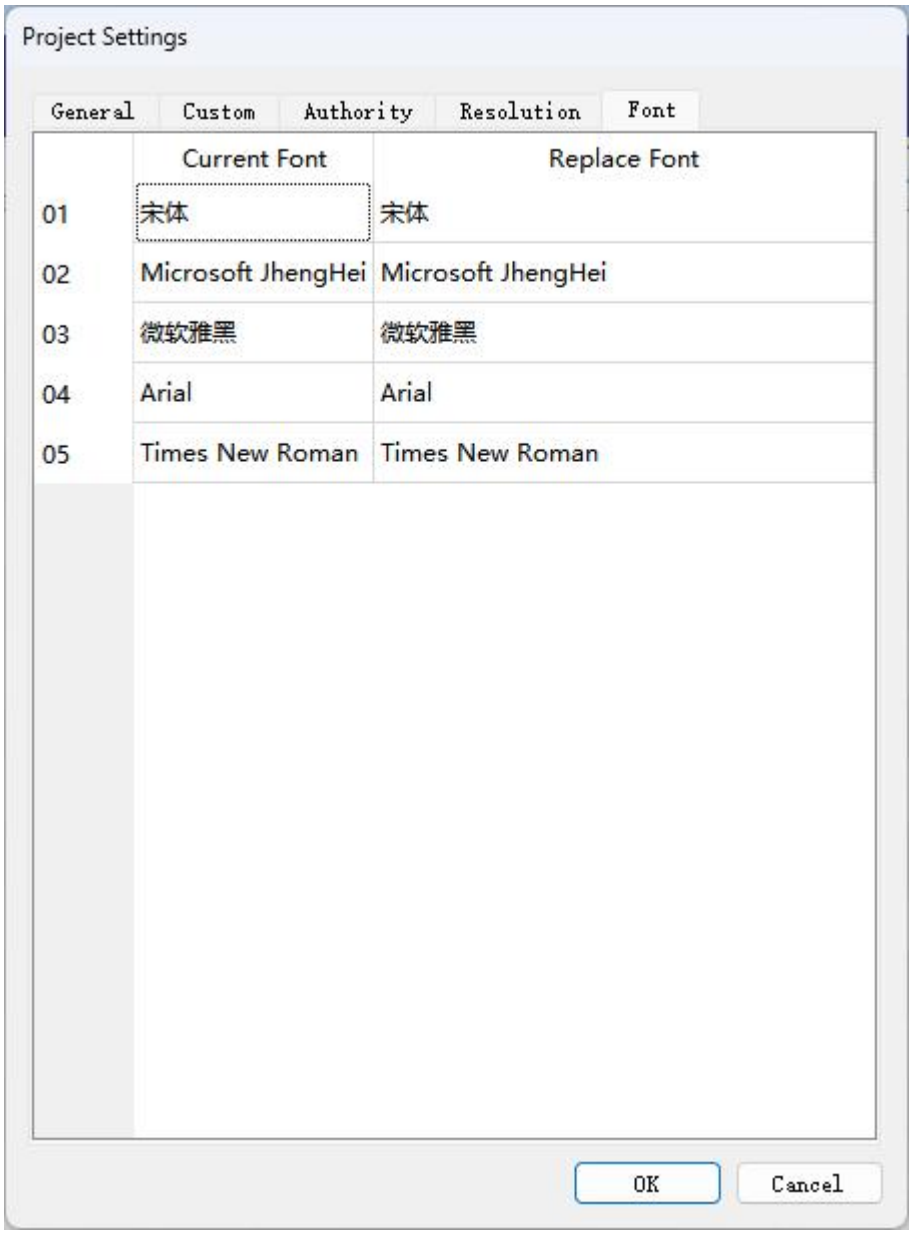



Figure 3.6.3-4 Setting Project Font Family

Once the project fonts are set, you can see that the font options in the control properties include the project fonts, as shown in Figure 3.6.3-5.

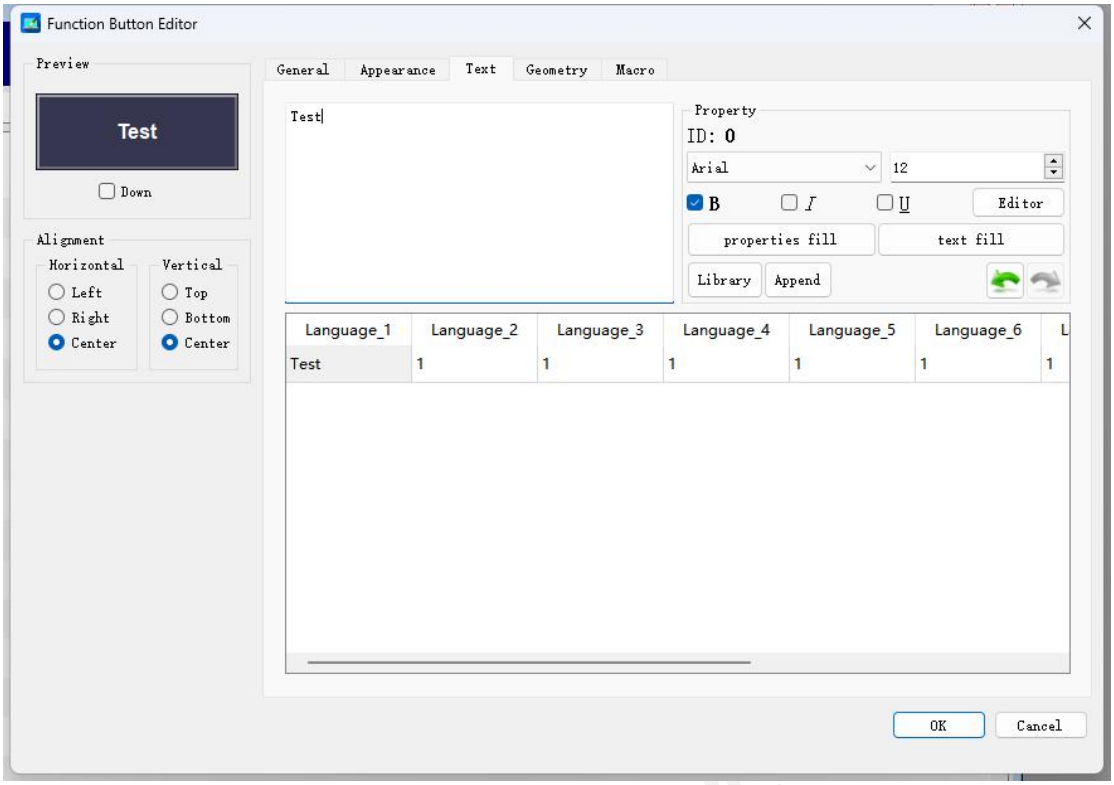
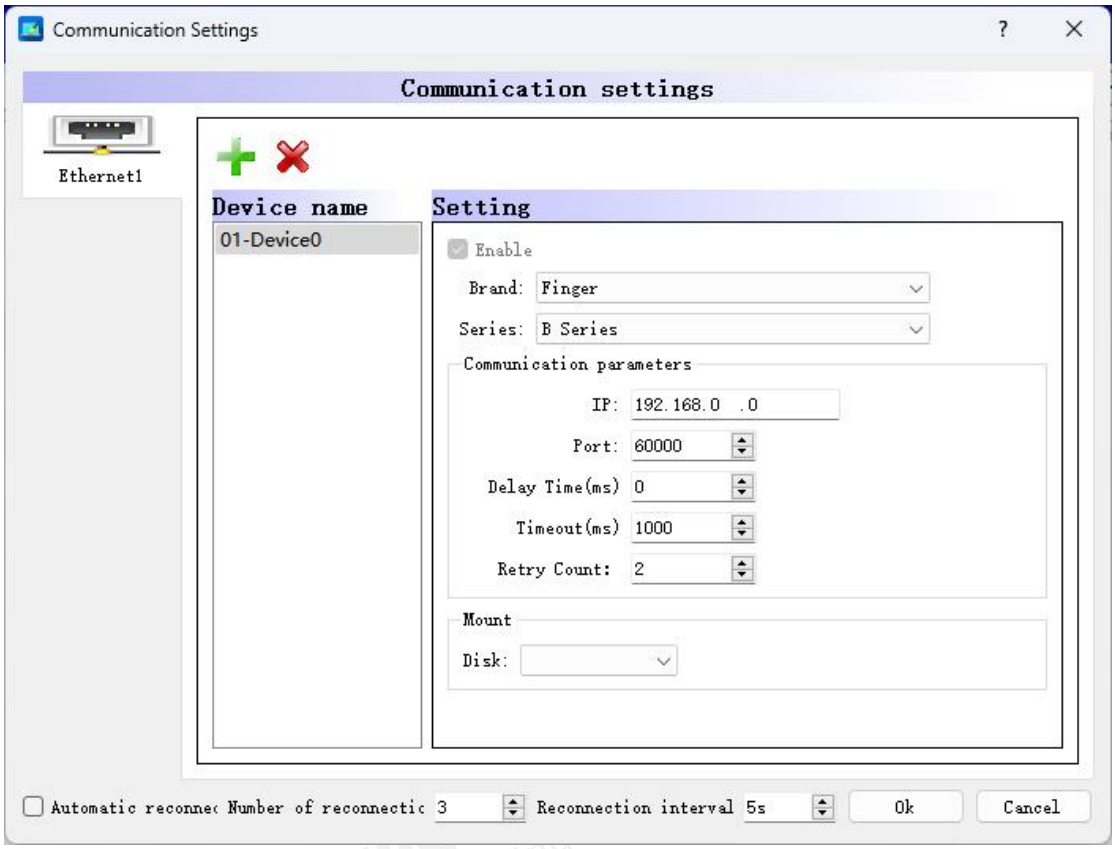



Figure 3.6.3-5 Font Properties in the Functional Button Plugin

## 3.6.4. Communication Settings




The communication settings allow you to mount files from the controller onto the PC. Combined with PCbase, this enables access to the controller's files from the PC.

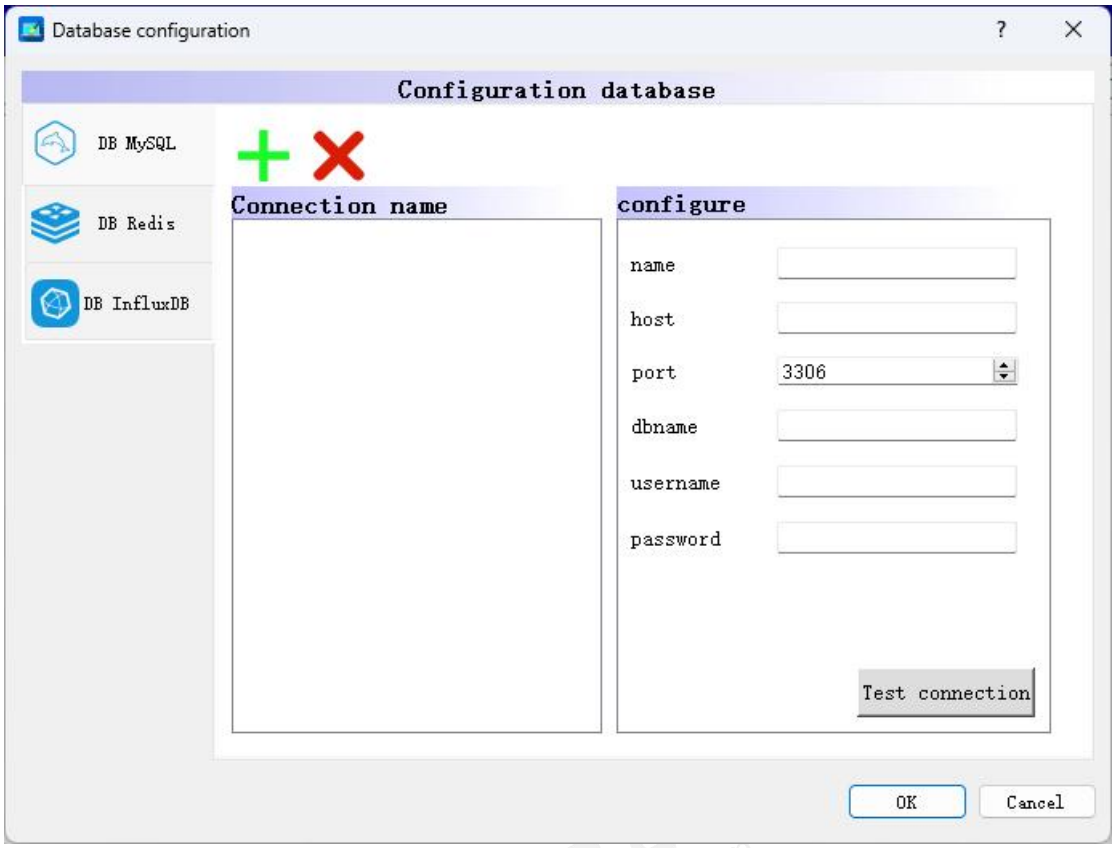
IP: The IP address of the controller, which should be in the same subnet as the PC.

Port: This should match the port set on the controller, usually 60000.

Mounted Drive Letter: Set to an unused drive letter on the PC.

## 3.6.5. IoT Settings

**1. Entry**

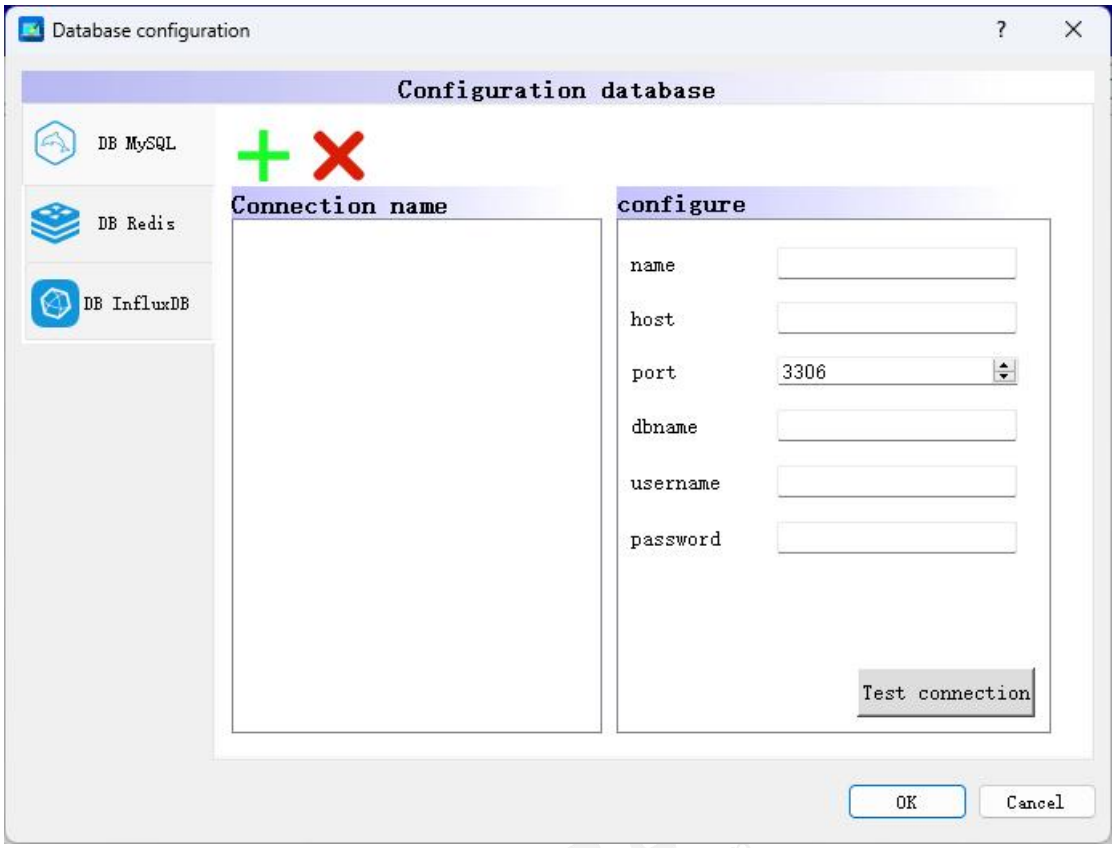



## 2. Function Overview

This setting is primarily used to configure the data source for IoT when the IoT functionality is enabled. It allows for the configuration of database data sources and supports MySQL, Redis, and InfluxDB, with MySQL currently being the only one in actual use.

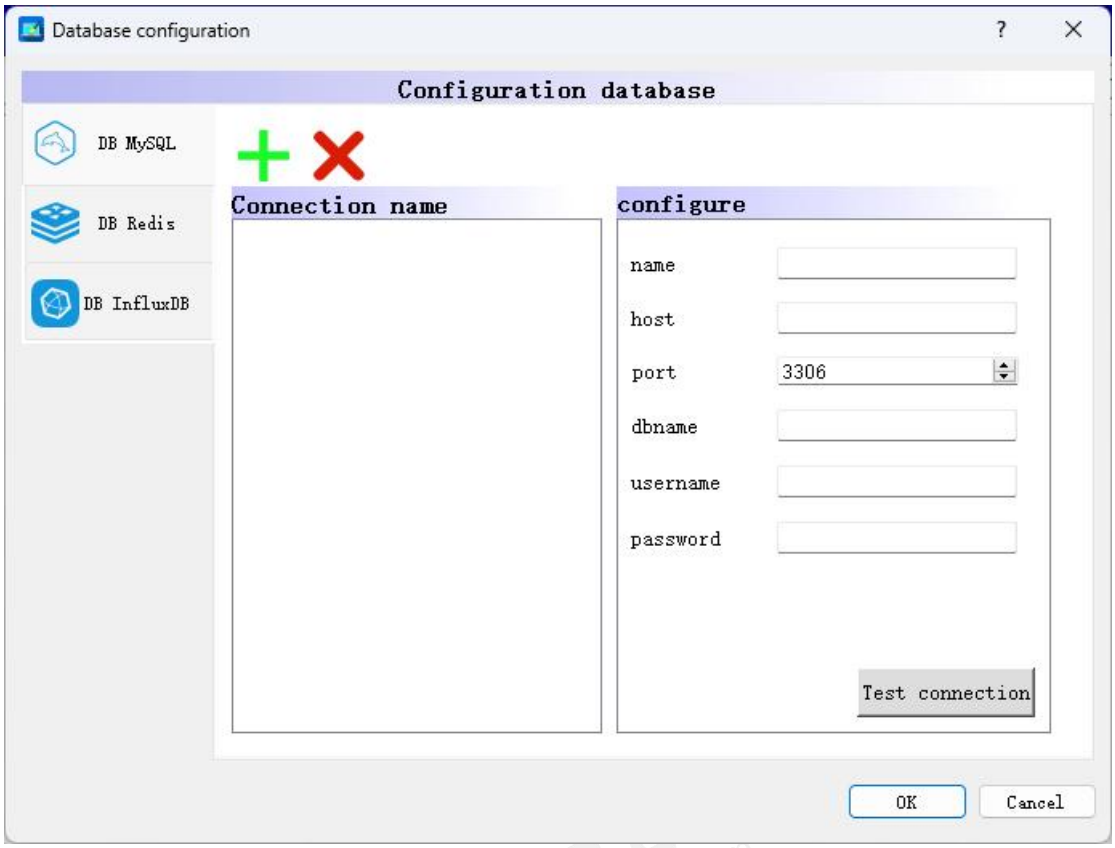
## 3. Database Configuration

### 1) Create Connection




As shown in the image above, click the + button to create a database connection. After creation, a connection will be added under the connection name column.

**2) Delete Connection**




As shown in the image above, select a connection under the connection name column, then click the button to delete the connection. Note that the deletion process varies depending on the connection's usage status:

- Connection Not in Use: If the connection to be deleted is not currently used in the HMI, it can be deleted directly.
- Connection in Use: Further actions may be required.

if the database connection to be deleted is already in use in the HMI, the deletion process will redirect to the "Convert Database Connection" settings window for conversion setup. this window will list the components associated with the database connection to be deleted. The user can then choose to replace the components that are linked to this database connection with another database connection, thereby achieving the conversion.

### 3) Configuration

- **Name:**

Represents the name of the database connection.

- **Host:**

Represents the database's host address, for example, 192.168.0.1.

- **Port:**

Represents the port number of the database, for example, 3306.

- **Database Name:**

Represents the name of the database, for example, cnc.

- **Username:**

Represents the username for the database, for example, root.

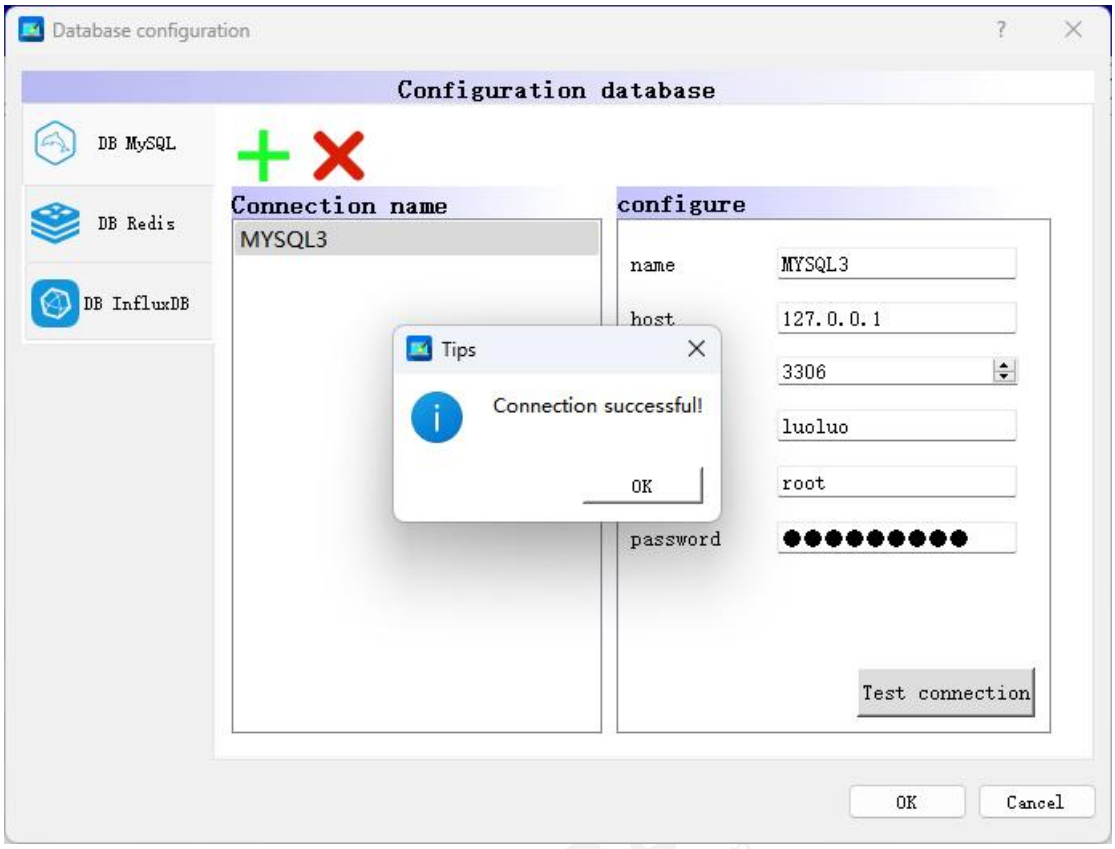
- **Password:**

Represents the password for the database connection, for example, 123456.

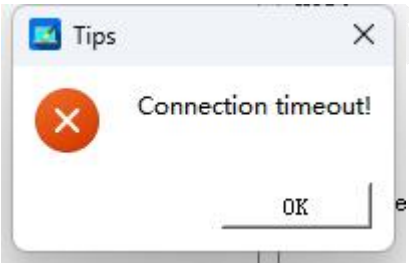
- **Test Connection:**

Tests the connectivity of the current database configuration. Possible prompts during connection testing.

- **Connection Successful**

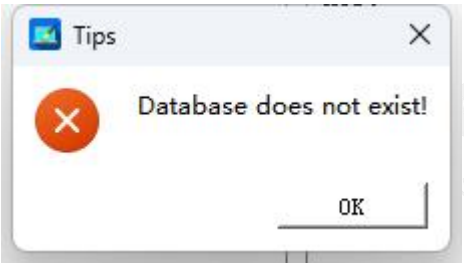



- **Connection Timeout**

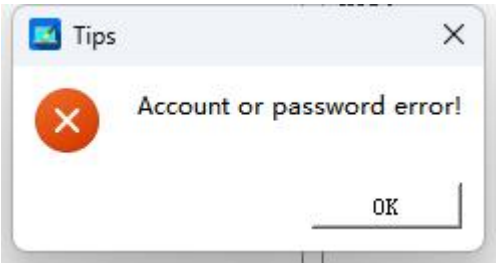



Indicates a timeout due to network issues or incorrect host address.

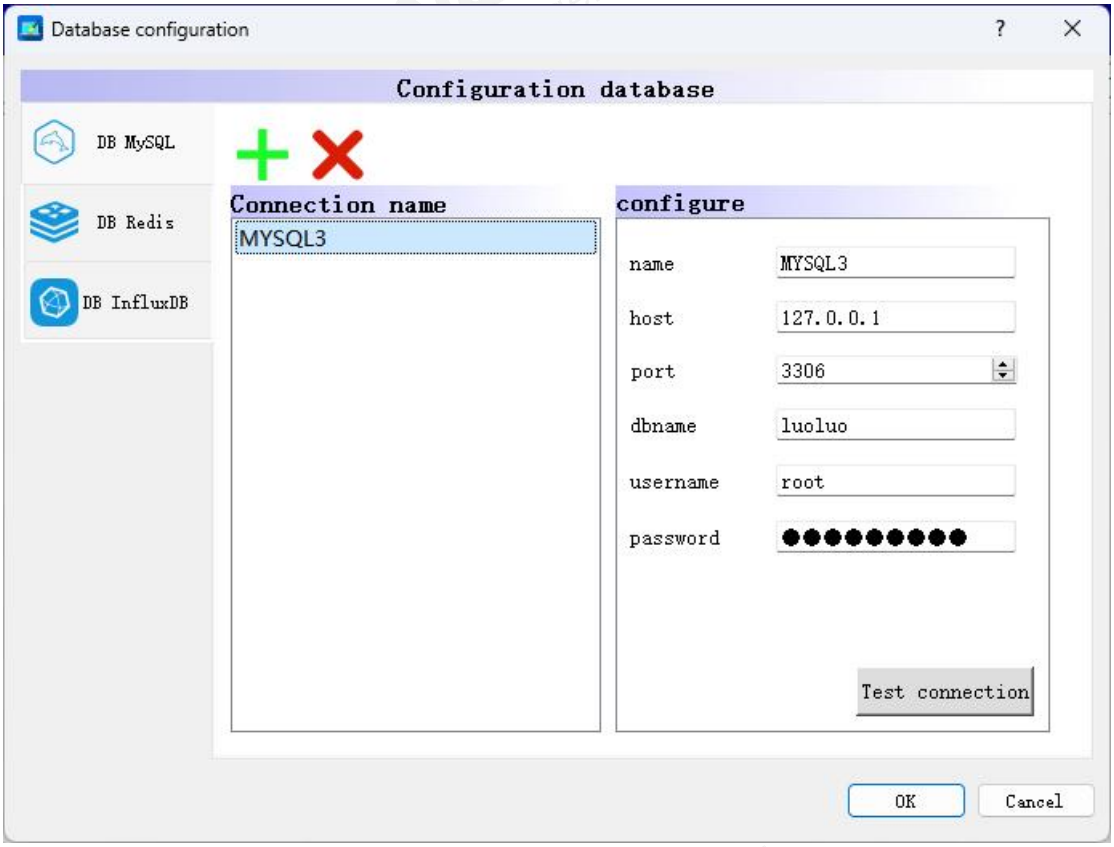
- **Database Name Incorrect**




- **Username or Password Incorrect**




### 4) Configuration Explanation




As shown in the above image, the database configuration screen consists of tab labels, buttons for adding and deleting connections, a list area, connection configuration area, and button groups

a. Tab Labels:

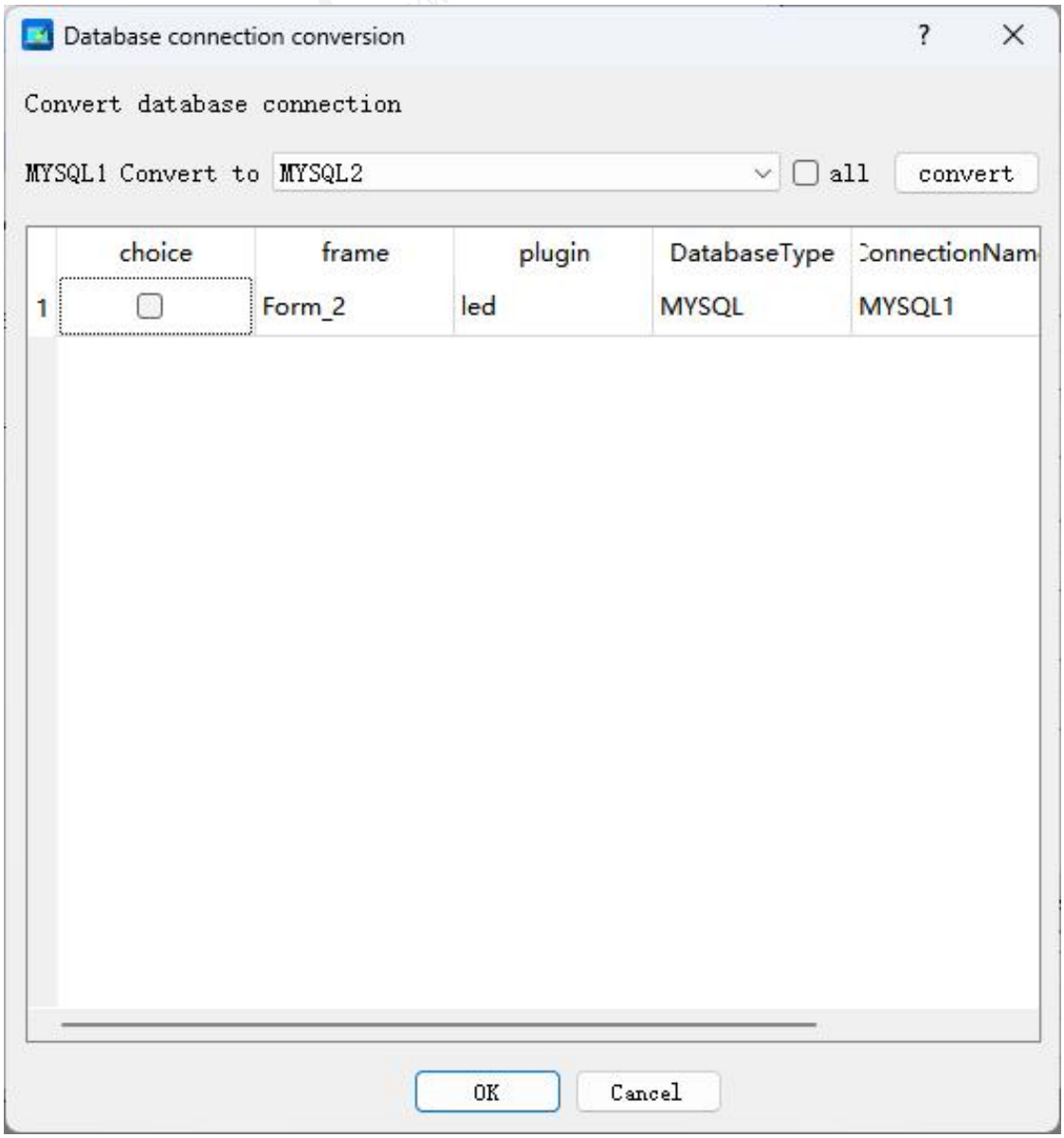
Switch between different databases by clicking on the tab labels. This will display the configured connection information for that database in the list area, configuration area, as well as the buttons for adding and deleting connections.

b. Add Connection Button:

Click this button to add a new connection. Once clicked, a new connection will be pre-generated, and its initial information will be displayed in the configuration area for editing. If the confirmation button in the button group is not clicked, the new connection will not be effective.

c. Delete Connection Button:

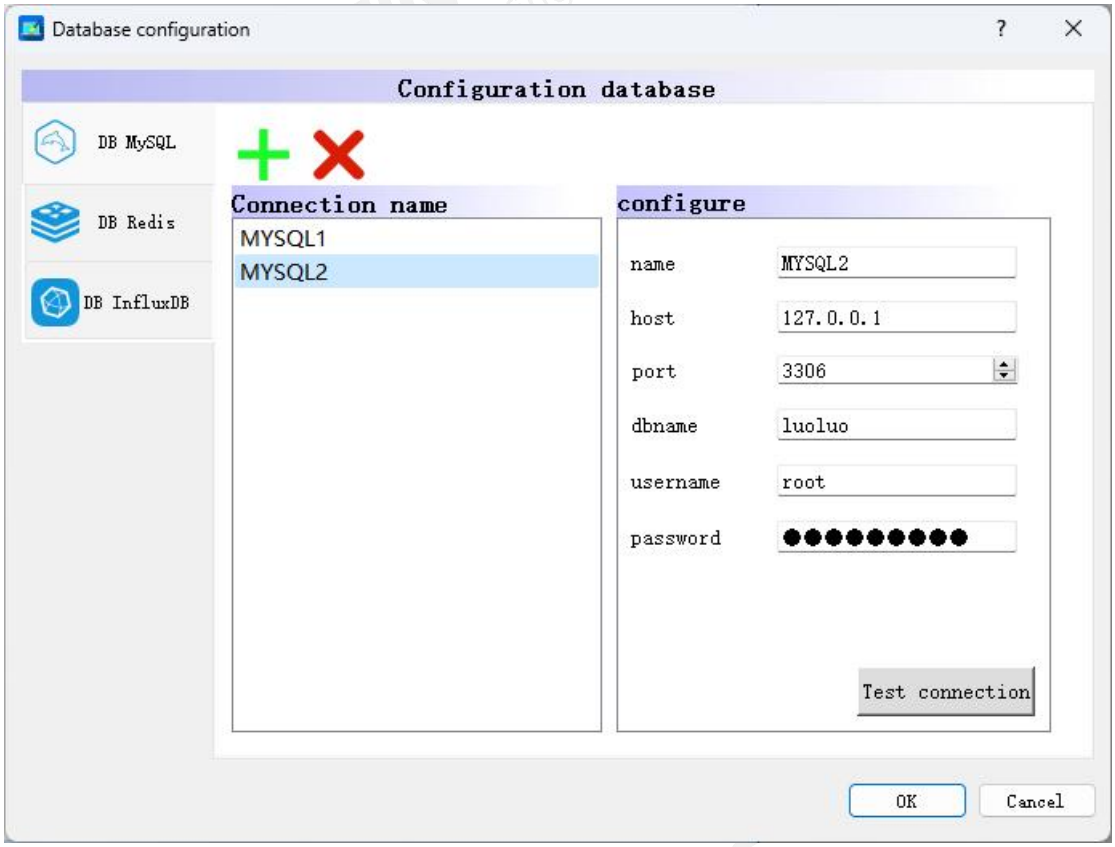
Click this button to delete a connection. Upon clicking, the system first checks if the connection is already associated with components in the project screen. If it is associated, a dialog will pop up to perform the conversion operation for the connected plugins.




In the dropdown menu, you can select an already configured connection. After checking

the target row, click 'Convert'. This action changes the connection name column of the selected row in the list. Finally, clicking the 'Confirm' button completes the conversion. If you do not click the confirmation button, the conversion will not take effect. Simultaneously, the database configuration screen's list area will remove this converted connection. Without clicking the confirmation button in the button group, changes and deletions will not be applied.

a. list area




As shown above, the connection information list area displays the configured connection details. When you click the "Add" button, newly generated connections are appended to this list. Specific details of the selected connection are displayed in the configuration area, where they can be modified. You can also test connectivity using the test button.

b. Configuration Area:

By selecting a connection from the list area, you can modify its specific details in the configuration area. Changes made while switching options in the list are saved to the cache. However, these changes only take effect when you click the "Confirm" button in the

button group; otherwise, they are not applied.

c. Button Group:

The button group includes "Confirm" and "Cancel". After modifying, adding, or deleting a connection configuration, clicking "Confirm" applies the changes. Clicking "Cancel" or directly closing the window without confirming will discard the current operation.

### 3.6.6. Properties

## 3.7. Windows

## 3.8. Help

# 4. Common Properties

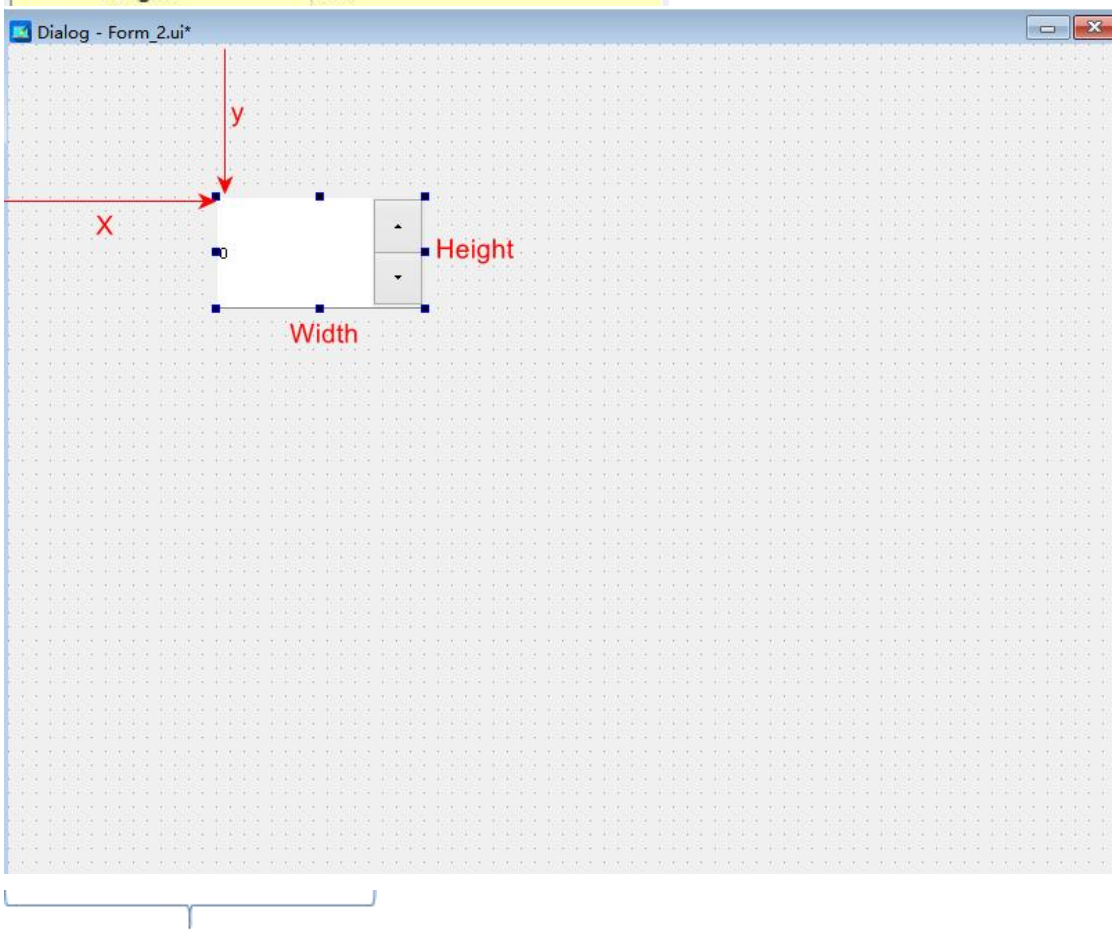
## 4.1. Object Name

The object name must start with a letter or underscore and can consist of letters, numbers, and underscores. Avoid using keywords as object names. Each object can be identified and accessed by setting its object name (Object Name), which is used to reference and manipulate objects in code.


## 4.2. Geometry Dimensions

Geometry dimensions consist of four properties: x, y, width, and height. Here, x and y denote the offset of the top-left corner of the control relative to the top-left corner of its parent window. Width and height specify the dimensions (width and height) of the control.

| geometry | [(670, 500), 30 x 30] |
|---|---|
| X | 670 |
| Y | 500 |
| Width | 30 |
| Height | 30 |




## 4.3. Font

| font | [Simsun, 9] |
|---|---|
| **Family** | 宋体 |
| **Point Size** | 9 |
| Bold | ☐ |
| Italic | ☐ |
| Underline | ☐ |
| Strikeout | ☐ |
| Kerning | ☑ |
| Antialiasing | PreferDefault |

- **Font Family**

Select a font family from the available fonts in the project. Available fonts can be set in the project settings.

- **Point Size**

Set the font size.

- **Bold**

Specify whether the font should be bold.


- **Italic**

Specify whether the font should be italicized.


- **Underline**

Specify whether the font should be underlined.


- **Strikethrough**

Specify whether the font should have a strikethrough.


[Letter Spacing Adjustment]

When checked, it effectively utilizes the gaps between character shapes, making the spacing between characters more uniform.

[Anti-aliasing]

Anti-aliasing, used to eliminate jagged edges that appear on the edges of objects in display output. Anti-aliasing is enabled by default. The image shows a comparison of anti-aliasing effects.

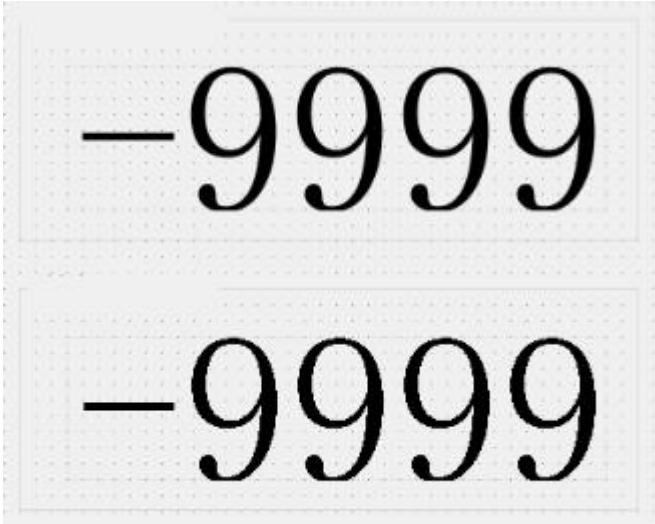



Figure 4.3-1 Anti-aliasing Comparison

## 4.4. Cursor

The appearance of the mouse pointer when it is moved over a component.

## 4.5. Focus Policy

The Focus Policy determines how a widget gains focus and handles focus events. The plugin provides different focus policies to suit various user

| Policy | Handling Method |
|---|---|
| NoFocus | The widget will not receive focus and will not handle focus events. This is typically used for display-only widgets or widgets that do not require user interaction. |
| TabFocus | The widget can receive focus using the Tab key. This is a common policy for most widgets that accept user input. |
| ClickFocus | The widget can receive focus when clicked with the mouse. This is useful for widgets that can be edited with a click. |
| StrongFocus | The widget can receive focus through the Tab key or mouse click. This is a more flexible focus policy, commonly used for custom user interfaces and special interaction needs. |

| | |
|---|---|
| WheelFocus | The widget can receive focus through the Tab key, mouse click, and mouse wheel events. This is useful for widgets that require scroll control. |

## 4.6. Layout Direction

| Policy | Handling Method |
|---|---|
| LeftToRight | Left-to-right layout |
| RightToLeft | Right-to-left layout |
| LayoutDirectionAuto | Automatic layout (default is left-to-right) |

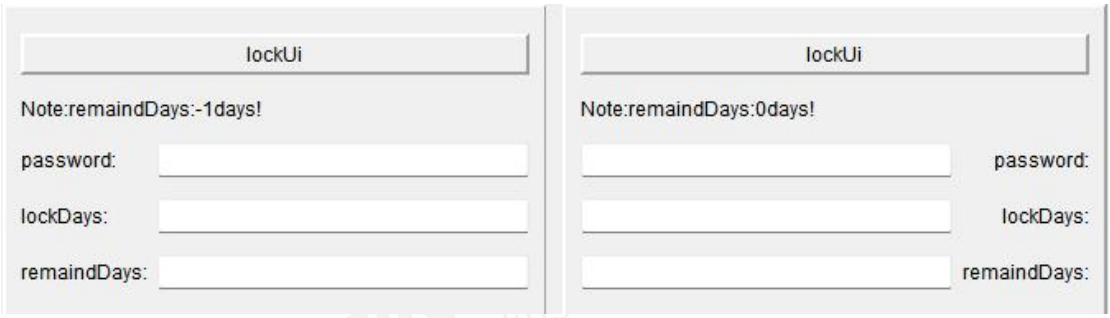



Figure 4.6-1 Layout Comparison

## 4.7. Auto Fill Background

When selected, the background color becomes effective; if not selected, the background color will not be effective, or the component's background color will follow the parent widget's background color.

## 4.8. Usage

Default checked. When checked, the plugin is enabled. When unchecked, the plugin is disabled.

## 4.9. Color Palette

Default option is "Inherit," which maintains the parent widget's color palette properties. Clicking "Inherit" changes the display to "Change Palette." Clicking this option opens the color palette editing window.

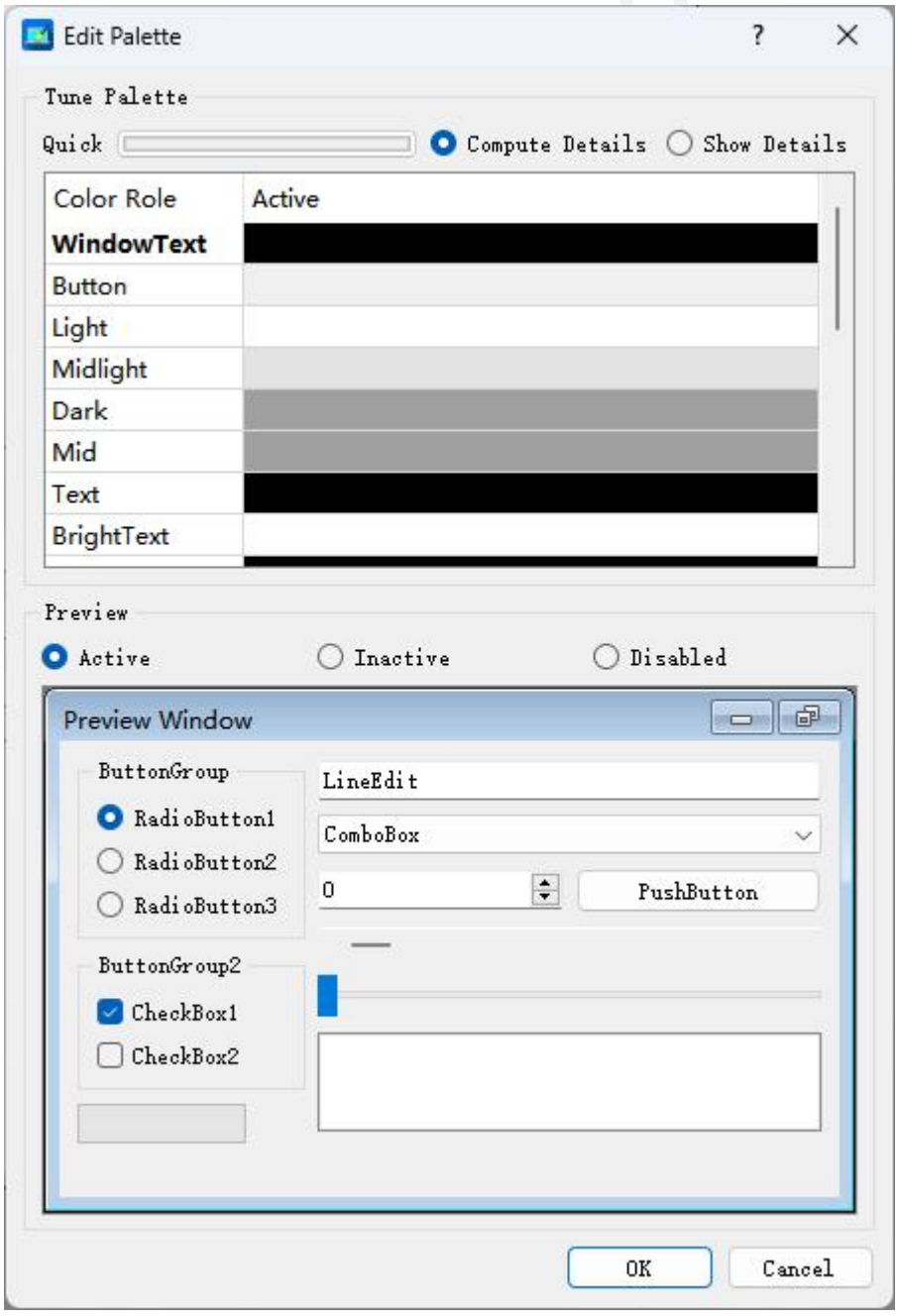



Figure 5.1-2 Edit Color Palette

Quickly: Click the white box next to the title to select a color. The computer will adjust the

active color of all dialogues based on the selected color (represented as RGB values). Selecting green quickly (120, 255, 255) has this effect.

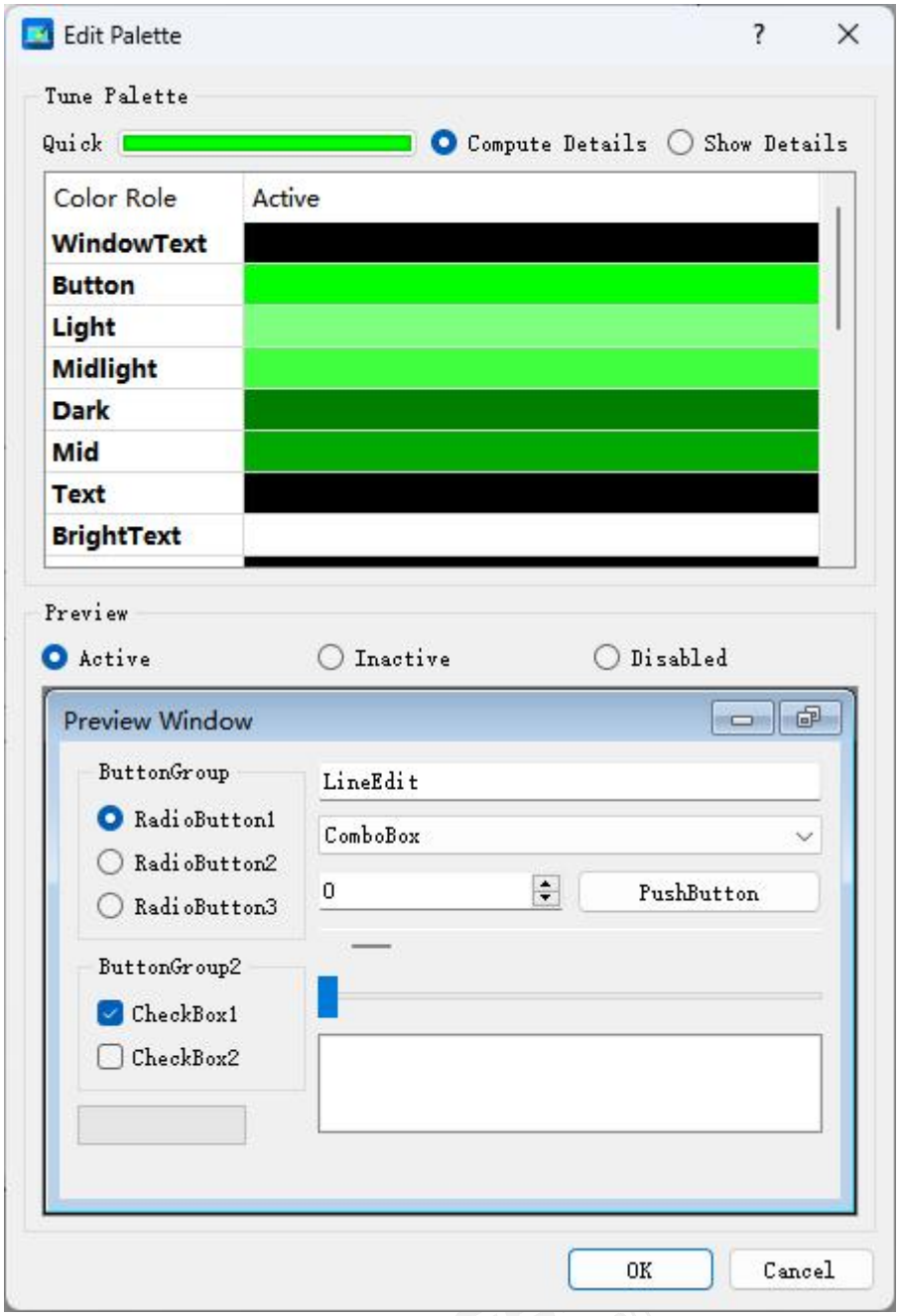



Figure 5.1-3 Quick Selection of Green Color

1. Automatic Adjustment: The computer automatically adjusts colors for two states of components: non-active and disabled.
2. Manual Adjustment: Users can individually adjust colors for three states of components: active, non-active, and disabled.

3. Preview: After setting up the color palette, users can select active, non-active, and disabled to preview the effects in these three states. "Active" refers to the currently displayed window, "non-active" refers to windows in the background without being minimized, and "disabled" indicates that the window cannot be edited.

## 4.10. Style Sheets

1. Style sheets allow for customizing plugin attribute styles. Attributes set in the parent component's style sheet will also affect similar attributes in child components. This functionality is not recommended for general user use.
2. For example, to change the color of the horizontal scroll bar's thumb to red, you can add a style sheet.

```
QScrollBar:horizontal {
height:30px;background:rgb(240,240,240);
margin:0px,0px,0px,0px;
padding-left:30px;
padding-right:30px;
}
QScrollBar::handle:horizontal {
background-color: red;
min-height: 30px;
}
```

Figure 4.10-1 Horizontal Scrollbar Style Sheet

# 4.11. Framework

| QFrame | |
|---|---|
| **frameShape** | Box |
| frameShadow | Plain |
| lineWidth | 1 |
| midLineWidth | 0 |

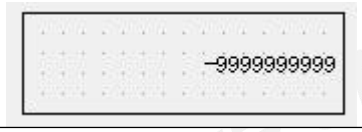
The framework's line width and centerline width parameters are illustrated below.


- **Border shape**

| Options | Explanation |
|---|---|
| NoFrame | No frame shape, meaning no display of a frame. |
| Box | Rectangular frame shape, where the component is surrounded by a simple rectangular border.   |
| Panel | Panel shape, similar to rectangular frame shape but with a raised or recessed effect. |

| | |
|---|---|
| |  |
| WinPanel | Windows-style rectangular panel, with a raised or recessed effect.<br>  |
| HLine | A horizontal line without a frame can serve as a separator.<br>  |
| VLine | A horizontal line without a frame can serve as a separator.<br>  |
| StyledPanel | The rectangular panel relies on the current GUI style, presenting either a raised or sunken effect.<br>  |

- **Border shadow**

| | |
|---|---|
| Plain | No shadow effect, displaying a plain panel or frame.<br>  |
| Raised | Raised shadow effect, appearing to float above the background.<br>  |
| Sunken | Sunken shadow effect, appearing to sink into the background. |

| | |
|---|---|
| |  |

- **Line Width**

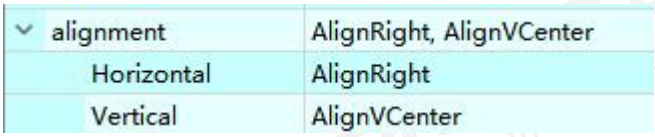
Sets the thickness of the border lines. Applies to Box, Panel, HLine, and VLine frame types. Setting line width for NoFrame, WinPanel, and StyledPanel will not produce a visual effect.

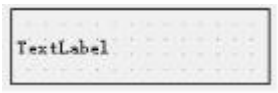
- **Center Line Width**

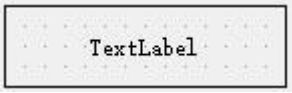
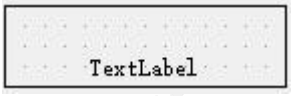
Sets the width of the center line of the border. Applies to the Raised and Sunken shadow effects.

## 4.12. Alignment

The text alignment strategy determines how text aligns within its container.




| Direction | Strategy | Effect |
|---|---|---|
| Horizontal | AlignLeft | Text aligned with the left edge of the container or page.  |
| | AlignHCenter | Text horizontally centered within the container or page.  |
| | AlignRight | Text aligned with the right edge of the container or page. |

| | | |
|---|---|---|
| | |  |
| | AlignJustify | Text justified within the container or page, aligning the left and right edges of the first and last lines, adjusting spacing between words to fit the width of the container or page.  |
| Perpendicular | AlignTop | Text aligned with the top edge of the container or page.  |
| | AlignVCenter | Text vertically centered within the container or page.  |
| | AlignBottom | Text aligned with the bottom edge of the container or page.  |

## 4.13. Access Control

1. Input-type plugins (such as numerical plugins, code editors, tables, etc.) can be configured with access control settings. The default access level for plugins is 0, meaning no access control is applied.
2. Human-Machine Interface (HMI) permissions are configured in the "Settings" ->

"Project Settings" -> "Access Control" page, in conjunction with access management plugins.

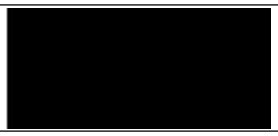
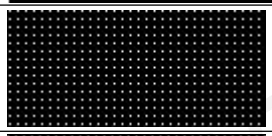
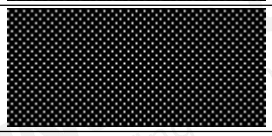
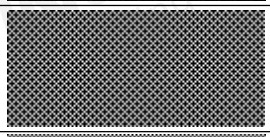
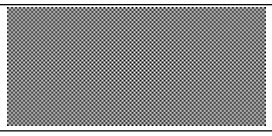
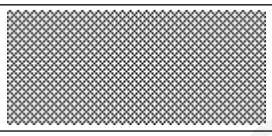
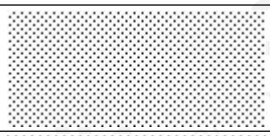
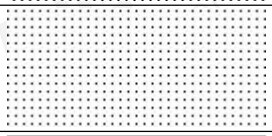
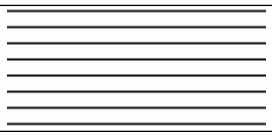
3. If the HMI permissions are greater than the plugin's permissions, access is denied. For example, if the required permission level to operate a plugin is 1, but the HMI permission level is 2, access will be denied due to insufficient permissions.

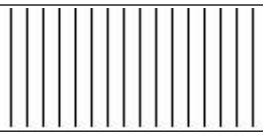
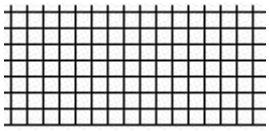
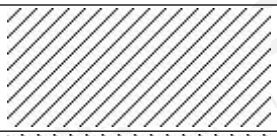
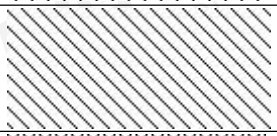
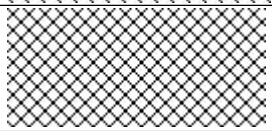
## 4.14. Line Types

| Option | Style |
|---|---|
| NoPen | No drawing |
| SolidLine |  |
| DashLine |  |
| DotLine |  |
| DashDotLine |  |

| | |
|---|---|
| DashDotDotLine |  |

## 4.15. Brush Style

| Option | Style |
|---|---|
| NoBrush | |
| SolidPattern |  |
| Dense1Pattern |  |
| Dense2Pattern |  |
| Dense3Pattern |  |
| Dense4Pattern |  |
| Dense5Pattern |  |
| Dense6Pattern |  |
| Dense7Pattern |  |
| HorPattern |  |

| | |
|---|---|
| VerPattern |  |
| CrossPattern |  |
| BDiagPattern |  |
| FDiagPattern |  |
| DiagCrossPattern |  |

## 4.16. Modal

Difference between modal and modeless:

Modal window: After it pops up, you cannot interact with other windows from the same application until it is closed.

Modeless window: After it pops up, you can interact with other windows from the same application simultaneously, even while it remains open.

## 4.17. Form Types

### 4.17.1. MainForm

The main form type, one of the most commonly used window types, primarily used for editing and housing plugins. The startup interface must be a main form type.

### 4.17.2. SubForm

The bordered sub-form type typically relies on a main form for use. The distinction

between bordered and borderless sub-forms lies in the display of the border. The title of the border can be edited using properties and macros.

### 4.17.3. FramelessSubForm

The frameless sub-form type lacks borders unlike its bordered counterpart.

### 4.17.4. KeypadForm

Keypad form type, mainly used for editing custom keyboards.

### 4.17.5. EmbeddedForm

**Embedded sub-screen type, used in conjunction with the Frame plugin.**

1. Frame adds the property referenceFormEnable, which determines whether the screen reference feature is enabled.
2. The Frame plugin adds the property autoZoomFormEnable, controlling whether the embedded sub-screen adapts its size.

1) When autoZoomFormEnable is enabled, the embedded sub-screen automatically adjusts its resolution to perfectly fit the Frame's geometry.
2) When autoZoomFormEnable is disabled, if the embedded sub-screen is larger than the Frame's geometry, scrollbars are used to adjust the display position. If the embedded sub-screen is smaller, the remaining space displays blank.

3. Nesting the use of Frame's reference feature is prohibited.

During editing, a popup prompts the user if nesting usage of Frame is detected.

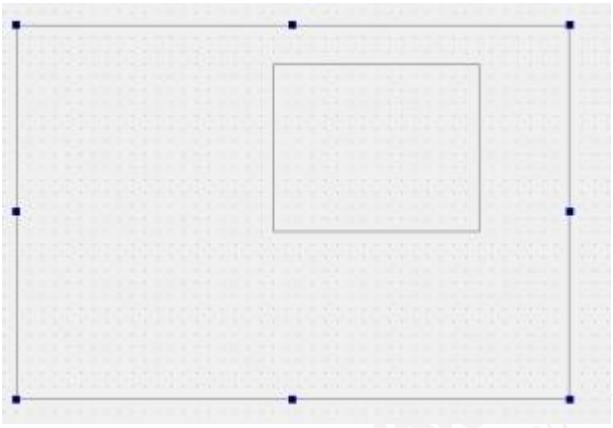


Figure 4.17.5-1 Does Not Support Nested Frames

4. When multiple Frames are linked to the same embedded sub-screen, only one Frame screen can display correctly.

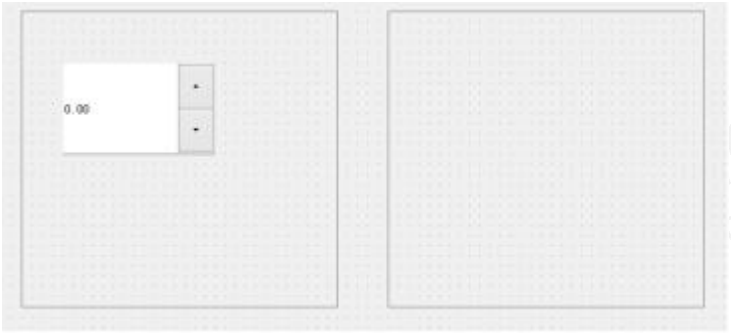



Figure 4.17.5-2 Embedded forms do not support reuse

5. In embedded subforms, the use of the Frame plugin's screen referencing feature is prohibited to avoid circular nesting. For instance, if subform A's Frame plugin references subform B, and subform B's Frame plugin in turn references subform A, it creates a loop of nested references which is not allowed.

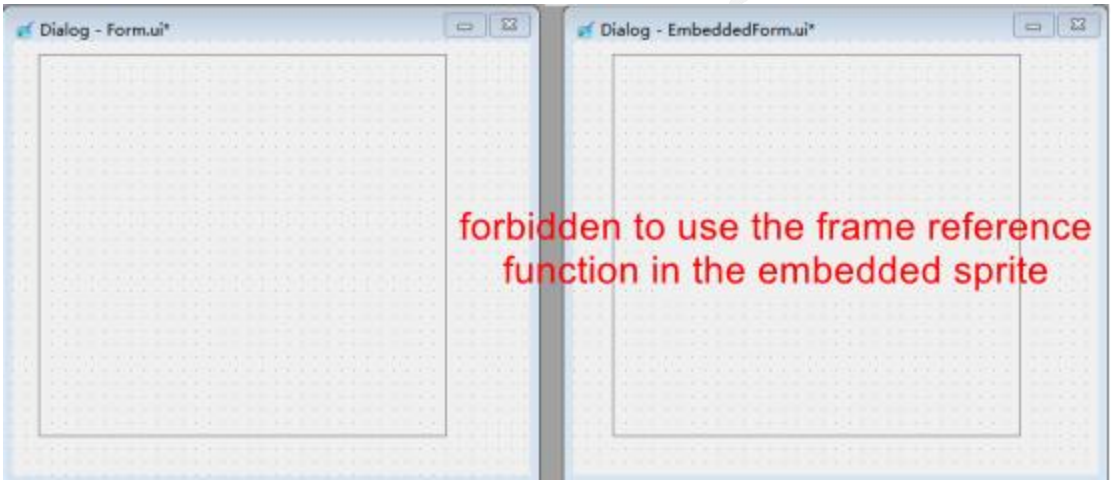


Figure 4.17.5-3: Embedded Sub-screen Prohibiting the Use of Frame's Screen Reference Function

6. Add interfaces to allow macro switching of referenced sub-screens.

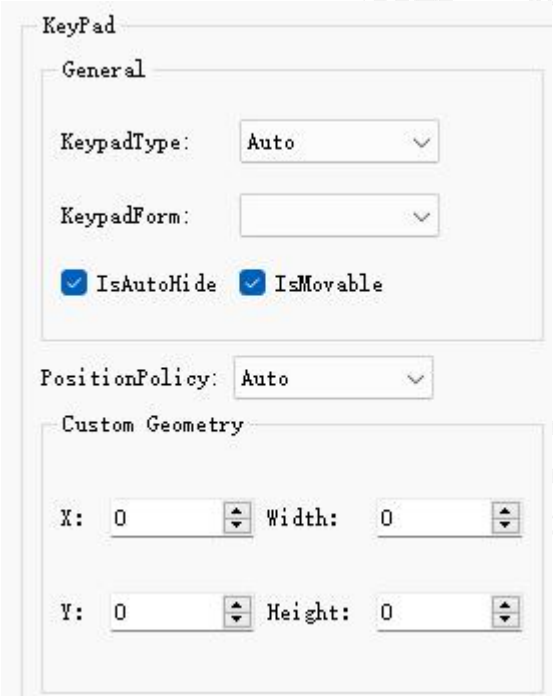
```
//Switching referenced screen
//name: Screen name
    bool switchReferenceForm(QString name);
```

Example：Form.frame.switchReferenceForm("Form_2");

# 5. Public Functions

## 5.1. Virtual Keyboard

### 5.1.1. Virtual Keyboard Property Editing




- **Keyboard Type**

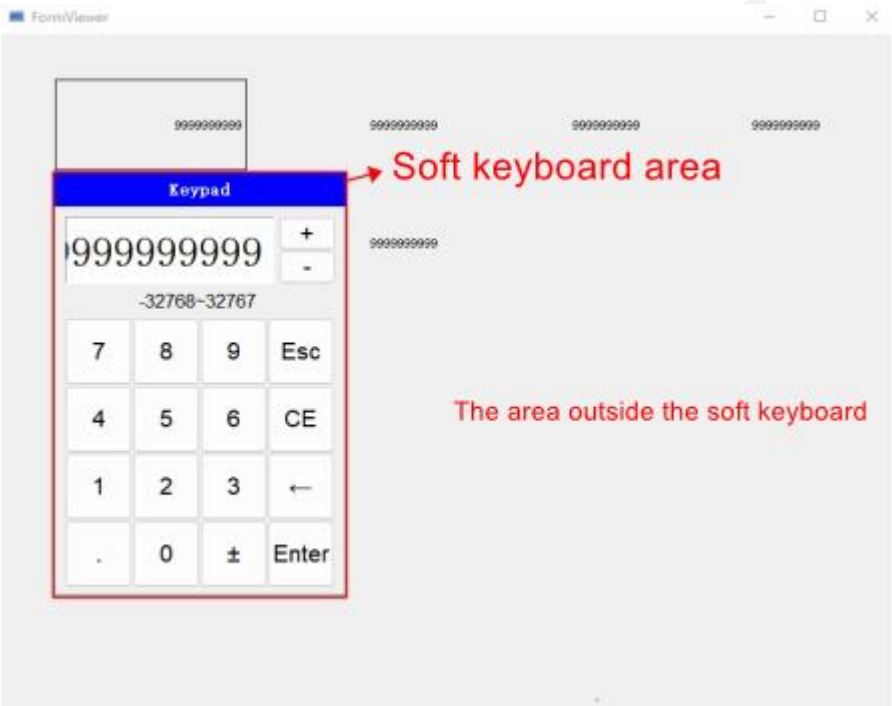
| Virtual Keyboard Type | Handling Method |
|---|---|
| None | Do not use virtual keyboard |
| Automatic | Use system virtual keyboard |
| Customize | Use Custom Virtual Keyboard |

- **Virtual Keyboard Screen**

When selecting custom virtual keyboard type, you need to choose a specific virtual keyboard screen to determine which custom virtual keyboard to use. (Virtual keyboard screen refers to screens with form type KeypadForm.)

- **Auto Hide**

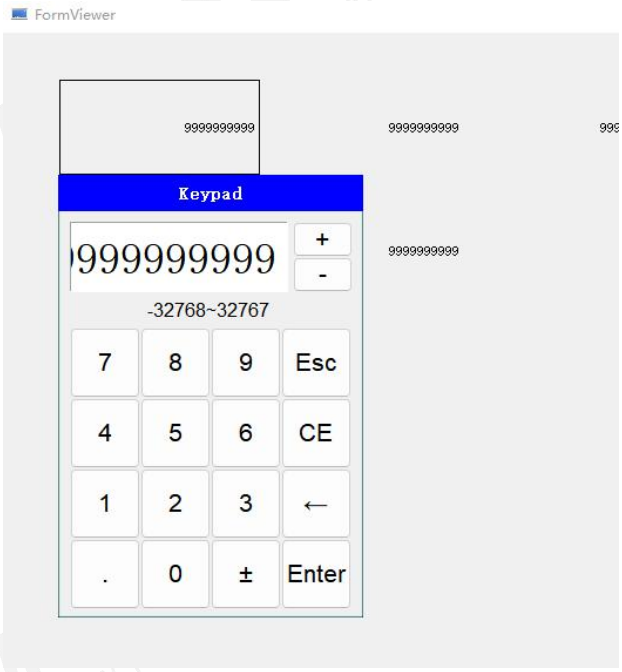
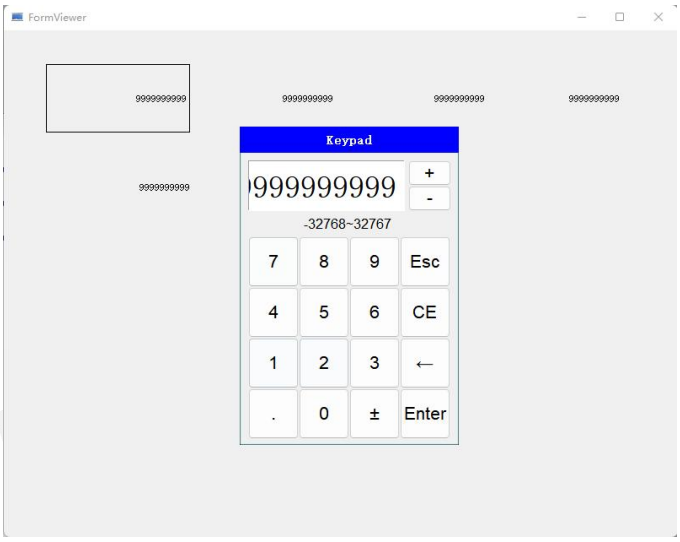
Enabling "Auto Hide" allows you to click outside the virtual keyboard area on the screen to cancel input and close the virtual keyboard.




- **Movable**

Enable "Draggable": When this option is enabled, the virtual keyboard can be moved by dragging its title bar.

- **Positioning Strategy**

| Strategy | Specific |
| --- | --- |
| Automatic | Automatically calculate the position of the virtual keyboard, initially displaying it below the plugin. If there's insufficient space below, attempt to position it above, to the right, or to the left.<br>  |
| Centered | Virtual keyboard displayed centered<br>  |
| 自 Custom | Display the virtual keyboard based on custom geometric dimensions' x and y coordinates. |

- **Custom Geometric Dimensions**

X: Horizontal coordinate of the top-left corner of the virtual keyboard

Y: Vertical coordinate of the top-left corner of the virtual keyboard

Width: Width of the virtual keyboard

Height: Height of the virtual keyboard

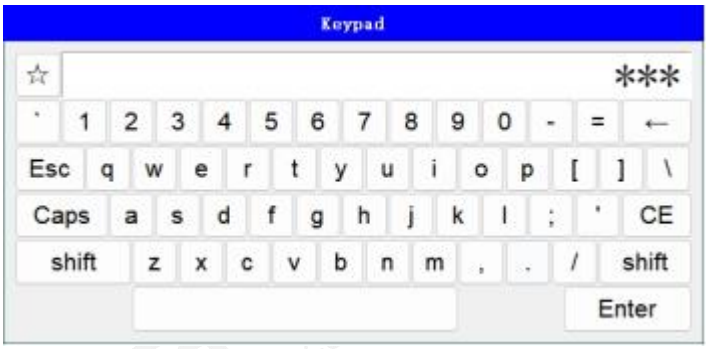
(Note: If either width or height is set to 0, default virtual keyboard dimensions are used.)

## 5.1.2. System Virtual Keyboard

The system virtual keyboard is as follows：

| Virtual keyboard type | Style |
|---|---|
| Binary |  |
| Octal |  |

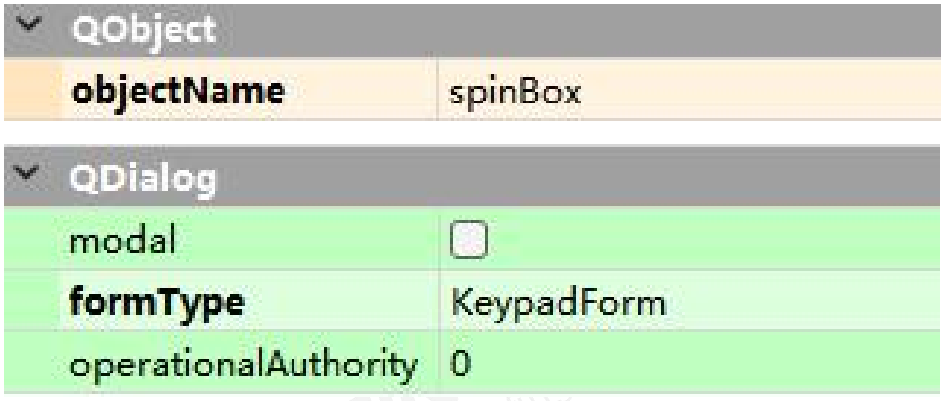
| | |
|---|---|
| Decimal |  |
| Hexadecimal |  |
| Numeric password input keyboard |  |

| | |
|---|---|
| Character password input keyboard |  |
| Chinese-English keyboard |  |

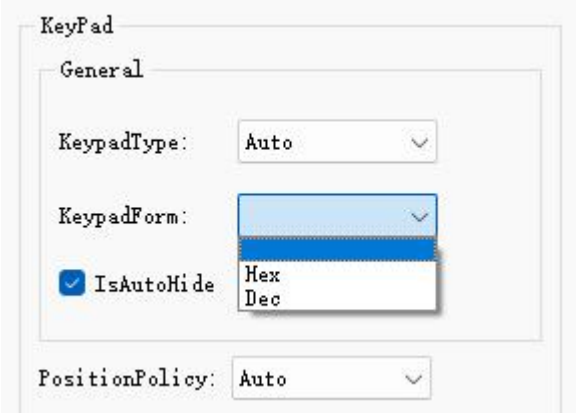
## 5.1.3. Custom Virtual Keyboard

### 1. Design Method

Create a new screen and set the form type in the property editor to "KeypadForm". The screen's object name will be the keyboard name.

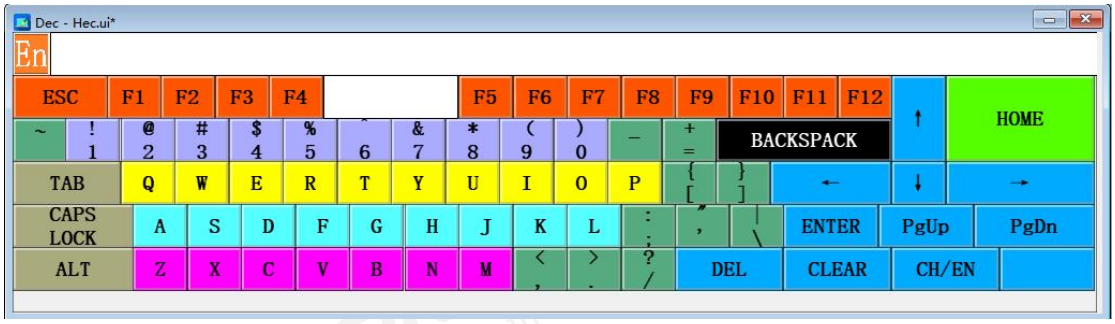



Once the above steps are completed, you can use the dropdown menu labeled "Virtual Keyboard Screen" to select the existing keyboard screen.




Custom virtual keyboards are designed by users and provide three components for design: "Keyboard Container," "Keyboard Keys," and "Keyboard Display."

**2. Example of a custom virtual keyboard.**



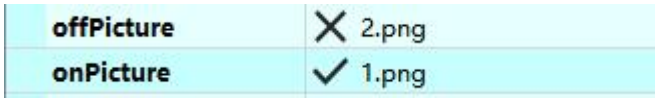

## 5.2. Multi-Language Editing

## 5.3. Variable Editing Popup

## 5.4. Image Resources

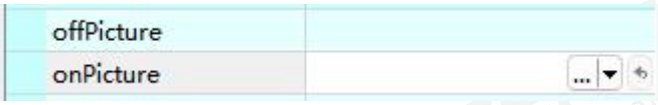
In the properties of icons and image types, you can use the "Image Resource Editor" to add image resources to the project.

**1. Properties of Icons and Image Types**

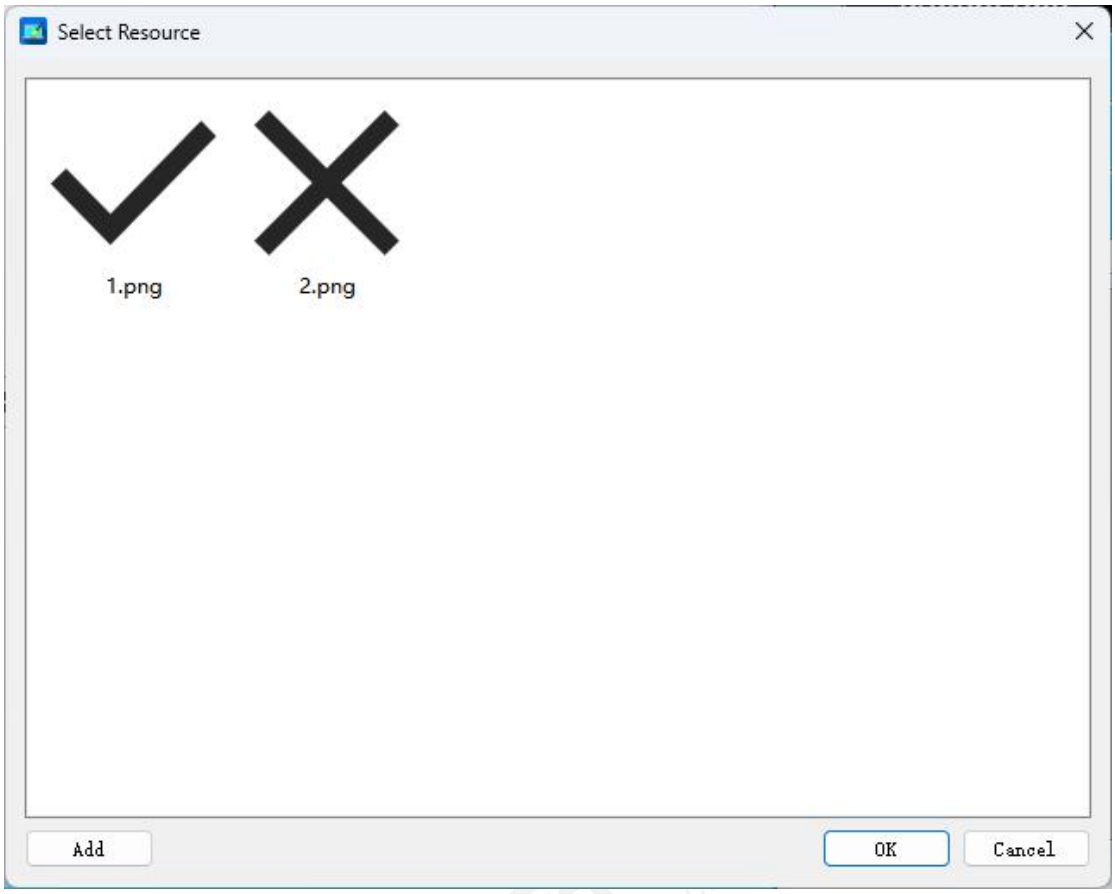



## 2. Image Resource Editor

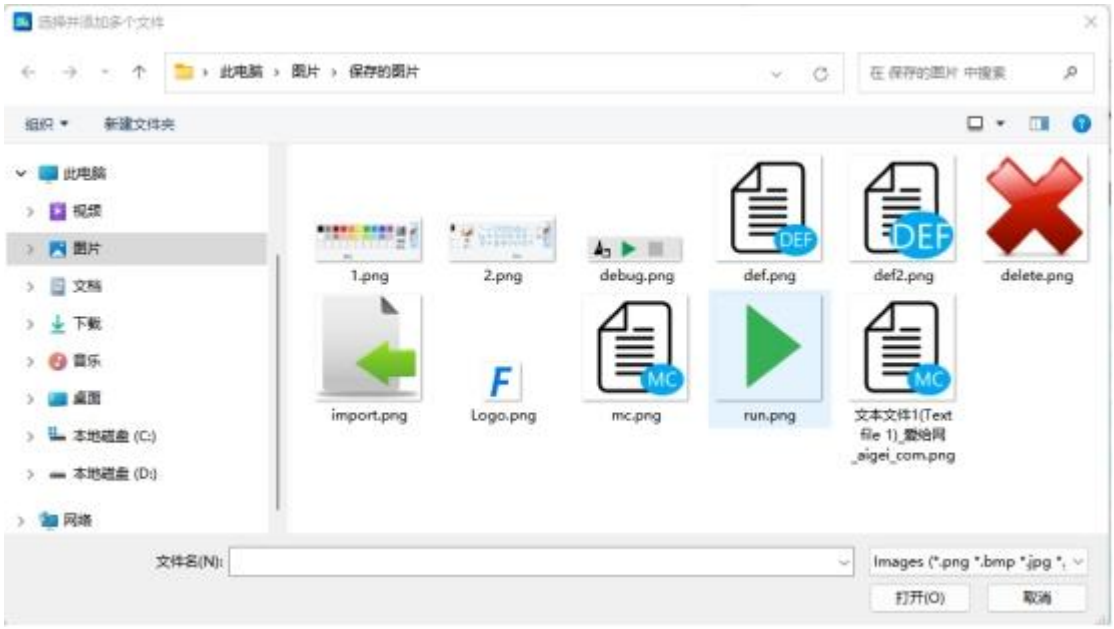
Select the property, click the "  " button on the right side.



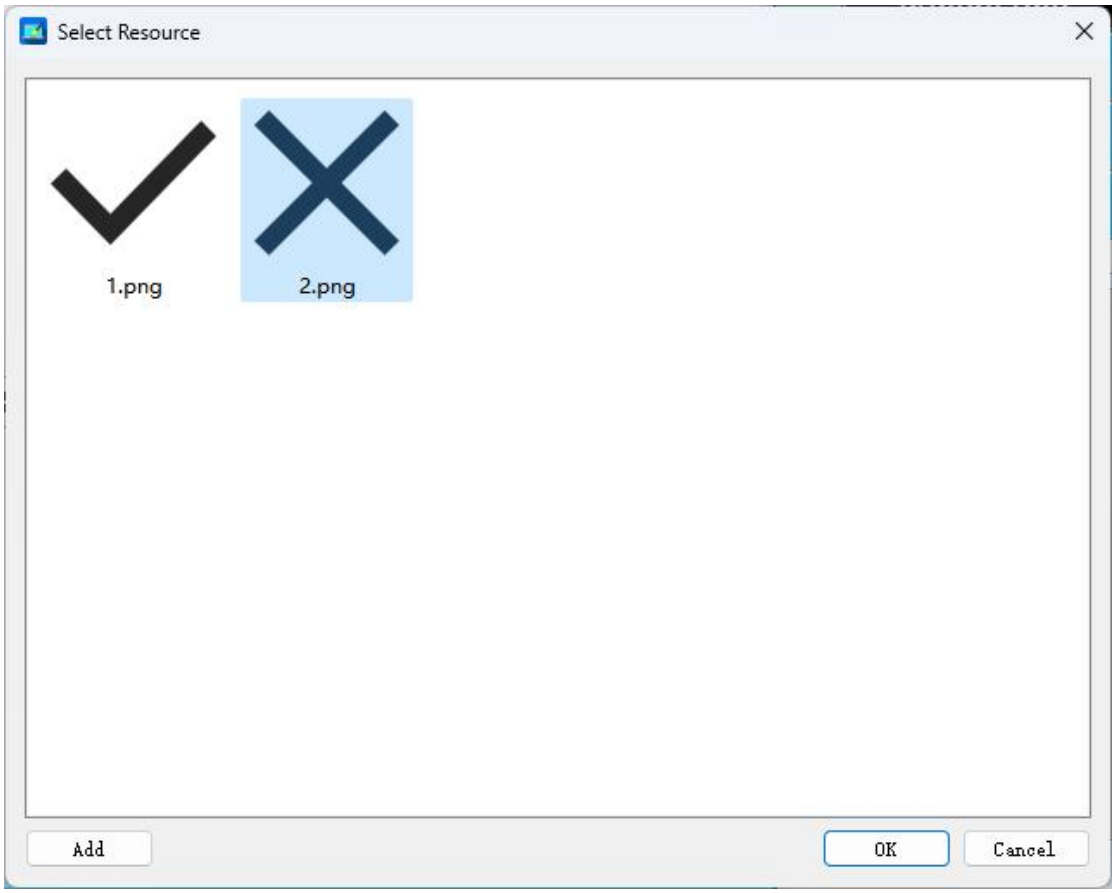

Open the Image Resource Editor.




Click the "Add" button and select the image you want to add from the local files.



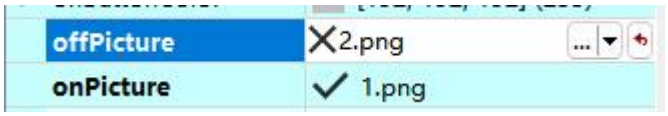

After adding, select the desired image.




Click "OK" to set it to the corresponding property.

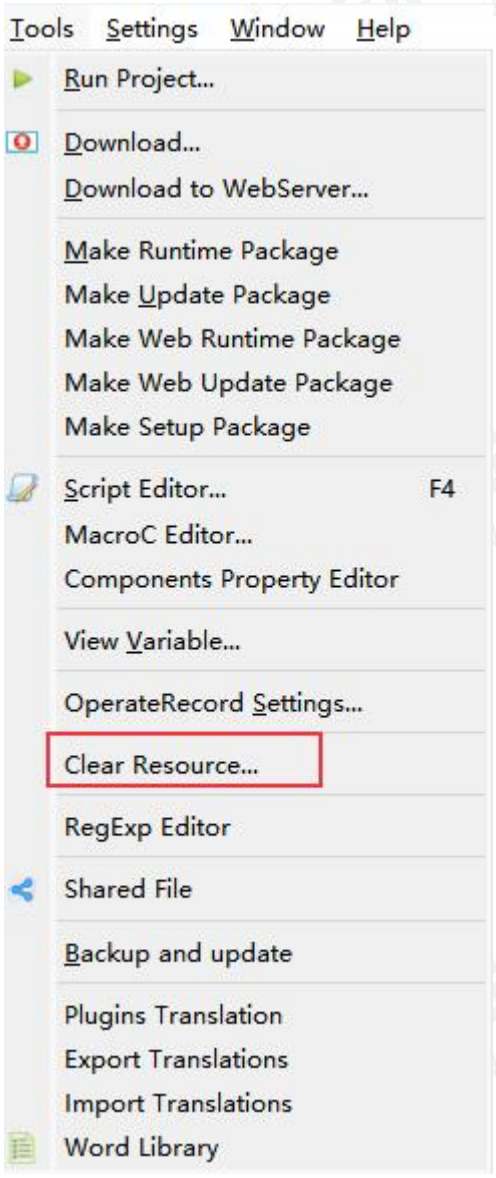
### 3. Reset Property Associated Image

Select the property and click the "  " button on the right side to reset the associated image for the property.

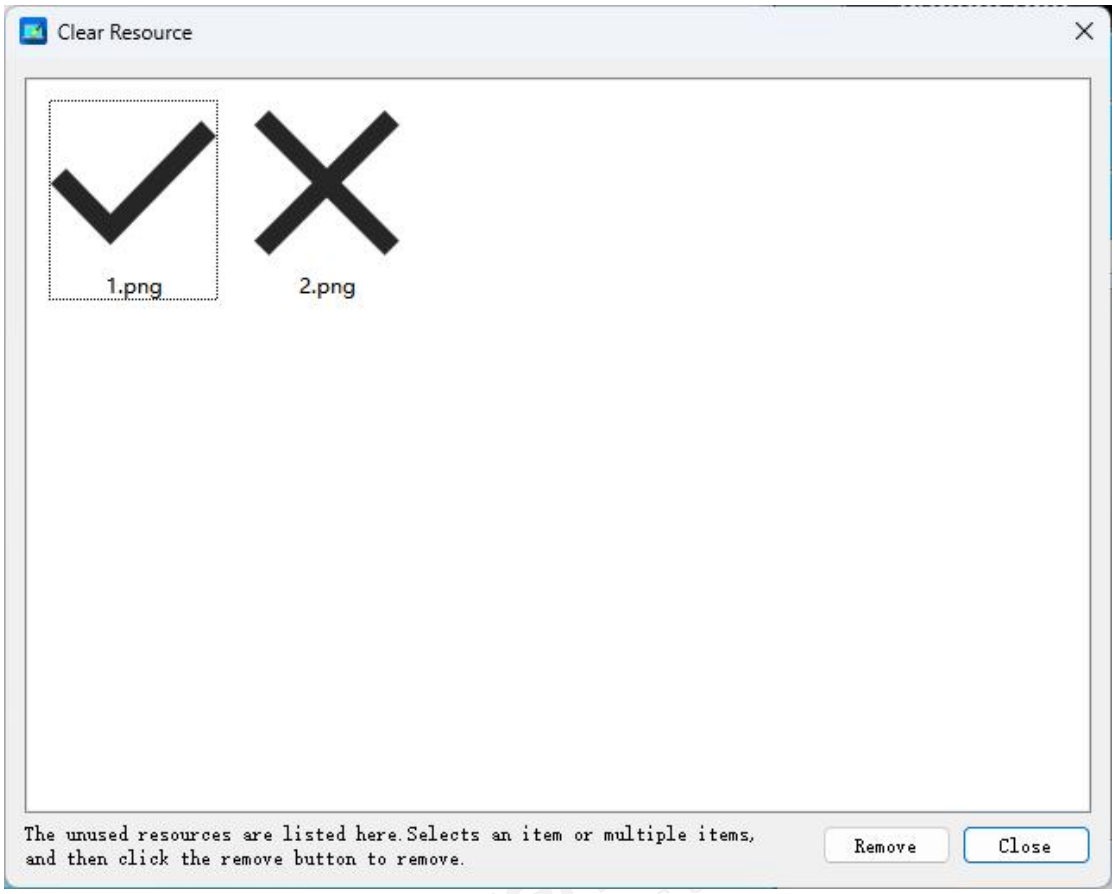



### 4. Cleaning Up Image Resources

The "Image Resource Cleaner" can be opened from the toolbar in the HMI software.




In the "Image Resource Cleaner," select the images not used in the project and click "Remove" to delete them from the project.




Note: Only images not used in the project will be displayed in the "Image Resource Cleaner" and can be removed. If an image has been used in the project (associated with a property), it cannot be removed.

# 6. Components

## 6.1. Horizontal Scrollbar

## 6.2. Vertical Scrollbar

## 6.3. Horizontal Slider

## 6.4. Vertical Slider

## 6.5. Time Editor

The Time Editor mainly provides users with an intuitive and convenient way to select and input time values or display time. This simplifies input operations in various application scenarios. When a time value is confirmed, the editor can send a time change signal, which external listeners can use to perform various data retrieval and processing actions. It is useful for projects that require data acquisition based on selected times.

### 6.5.1. Component Properties

[Object Name]

Default Name: dateTimeEdit

[Geometric Dimensions]

[Palette]

[Font]

[Cursor]

[Focus Policy]

[Layout Direction]

[Auto Fill Background]

[Usage]

[Alignment]

Refer to the common property descriptions for details.

## 6.5.2. QDateTimeEdit

[Maximum Date]

Specifies the maximum selectable date limit for the component.

[Minimum Date]

Specifies the minimum selectable date limit for the component.

[Format]

Specifies the format of the displayed time string, for example: yyyy/M/d H:m

[Date Time]

Specifies the current date and time for the component.

[Popup Calendar Time Selection Mode]

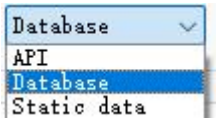
Specifies the time setting mode for the component. Checking this option indicates that clicking button 

during HMI runtime will pop up a calendar mode for selecting date and time. Otherwise, it will set the date and time using mode . 


## 6.5.3. IoT

This section is mainly used to set properties related to IoT.

[Data Source Type]

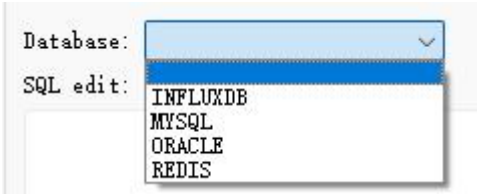



Used to set the data source type for IoT.

[Enabled]

Indicates whether IoT functionality is enabled.

[Database]




Select the database type to use from the dropdown list. Currently, only MySQL database is supported. This setting is displayed only when "Database" is chosen in the "Data Source Type" dropdown.

[Connection]

Select the database connection from the dropdown list. The dropdown list data is sourced from the IoT settings in the database connection configuration. This setting is displayed only when "Database" is chosen in the "Data Source Type" dropdown.

[SQL Editor]

Used to edit SQL query statements, for example: select col, col2 from table. This setting is displayed only when "Database" is chosen in the "Data Source Type" dropdown.

[Request Method]

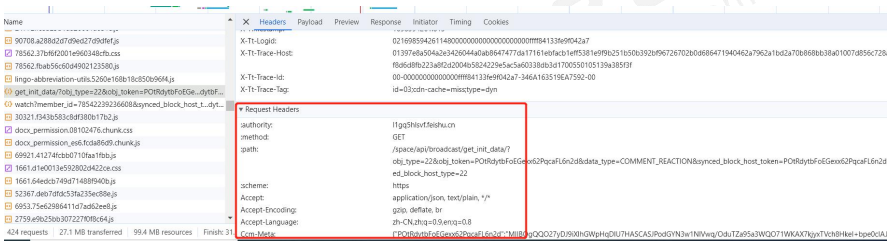
Used to set the request method for API requests. This setting is displayed only when "API" is chosen in the "Data Source Type" dropdown.

[URL]

Used to set the API request URL, for example: https://www.example.com. This setting is displayed only when "API" is chosen in the "Data Source Type" dropdown.

[Request Data Header]

Used to set the data header for API requests in key-value format, for example:



Add by clicking the button, delete by clicking the button. This setting is displayed only when "API" is chosen in the "Data Source Type" dropdown.

[Enable Mapping]

Set whether to enable data field mapping. If mapping is not enabled, the component requires the data field to be "date" (refer to the mapping table below). Otherwise, if the data is not mapped, it will not be successfully set in the component.

[Mapping Table]

When mapping is enabled, if the data field obtained from the database query is not "date," the user can map the queried data field to the "date" field.

[Data Converter]

Select the data converter to use from the dropdown list. The data items in the dropdown list are sourced from the data converters created in the "Edit" section below.

[Edit]

Manage data converters by clicking the Edit button.

[Preview]

Open the data preview window by clicking the Preview button.

[Auto Update]

Indicates whether the component is enabled to periodically fetch IoT data for automatic refresh.

[Update Frequency]

Indicates the periodic refresh cycle for the component's data, in seconds (s).

## 6.5.4. Component Macro Interface

1. Select Date

```
/**
* @Description: Retrieves the date when the time selector's value changes.
* @Return Value: Date format string [yyyy/MM/dd]
*/
```

Example：Form.date Time Edit.select Date()

2. Select Date Time

```
/**
* @Description: Retrieves the date and time when the time selector's value changes.
* @Return Value: Date and time format string [yyyy/MM/dd HH:mm]
*/
```

Example：Form.dateTimeEdit.selectDateTime()

3. Select Time

```
/**
* @Description: Retrieves the time when the time selector's value changes.
* @Return Value: Time format string [HH:mm]
*/
QString selectTime();
```

Example：Form.dateTimeEdit.selectTime()

4. Set Select Date

```
/**
* @Description: Sets the date of the time selector.
* @Parameter:
    • date: Date format string [yyyy/MM/dd]
* @Return Value: None
*/
void setSelectDate(const QString &date);
```

Example：Form.dateTimeEdit.setSelectDate("2022/06/08")

5. Set Select Date Time

```
/**
* @Description: Sets the date and time of the time selector.
* @Parameter:
    • dateTime: Date and time format string [yyyy/MM/dd HH:mm]
* @Return Value: None
*/
```

Example：Form.dateTimeEdit.setSelectDateTime("2022/06/08 13:13:01")

6. Set Select Time

```
/**
* @Description: Sets the time of the time selector (default date is current date).
* @Parameter:
    • time: Time format string [HH:mm]
* @Return Value: None
*/
void setSelectTime(const QString &time);
```

Example：Form.dateTimeEdit.setSelectTime("13:13:01")

7. Update IOT Data

```
/**
* @Description: Manually refreshes IoT data.
*/
void updateIOTData();
```

Example：Form.dateTimeEdit.updateIOTData()

## 6.5.5. Component Macro Properties

| Attributes | Types | Descriptions | Examples |
|---|---|---|---|
| date | QDate | Select date, referencing time settings in JavaScript's | Form.dateTimeEdit.date=new Date() |

| | | | |
|---|---|---|---|
| | | Date. | |
| dateTime | QDateTime | Select date and time, referencing time settings in JavaScript's Date | Form.dateTimeEdit.dateTime=new Date() |
| displayFormat | QString | Format time format | Form.dateTimeEdit.displayFormat=”yyyy/M/d H:mm” |
| enabled | bool | Plugin active status | Form.dateTimeEdit.enabled=true |
| focusPolicy | FocusPolicy | Focus strategy，NoFocus=0，TabFocus=1，ClickFocus=2，StrongFocus=11，WheelFocus=15 | Form.dateTimeEdit.focusPolicy=1 |
| font | QFont | Font | var fontMap={};<br>fontMap["weight"]=12;<br>fontMap["pointSize"]=12;<br>fontMap["family"]="宋体";<br>fontMap["italic"]=true;<br>fontMap["underline"]=true;<br>Form.dateTimeEdit.font = HProperty.setQFont(fontMap) |
| geometry | QRect | Geometry dimensions | var geoMap={};<br>geoMap["x"]=12; |

| | | | |
|---|---|---|---|
| | | | geoMap["y"]=12;<br>geoMap["width"]=12;<br>geoMap["height"]=12;<br>Form.dateTimeEdit.geometry = HProperty.setQRect(geoMap); |
| maximumDate | QDateTime | Maximum date, referring to time settings in JavaScript's Date. | Form.dateTimeEdit.maximumDate=new Date(7999, 11 ,31); |
| minimumDate | QDateTime | Minimum date, following JavaScript's Date time setting. | Form.dateTimeEdit.minimumDate=new Date(1752, 8, 14); |
| popupCalendar | Bool | Popup calendar selection mode | Form.dateTimeEdit.popupCalendar=true |
| visible | bool | Plugin visibility | Form.dateTimeEdit.visible=false |

## 6.5.6. Component Signals

Select Date Time Changed（Date and time selection change signal）

```
/**
* @Description: Signal for when the time selector's value changes.
* @Parameter:

    • dateTime: Date and time format string

*/
```

Indicates that this signal will be sent when the component's date and time change.

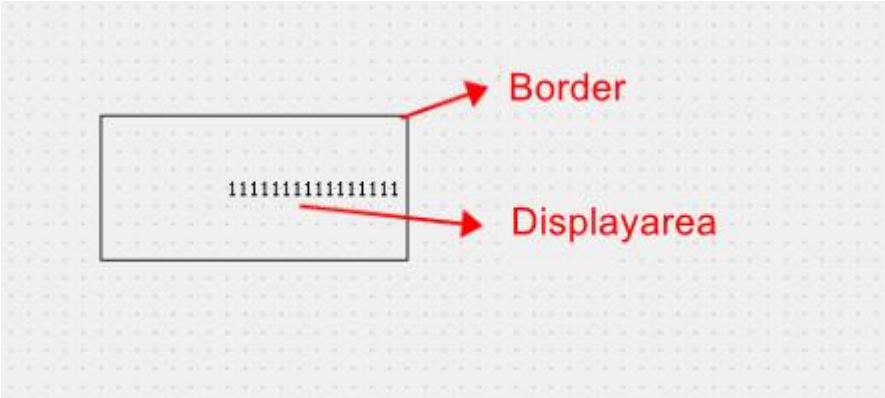
## 6.6. Input Block

## 6.7. Numeric

### 6.7.1. Function Introduction

This component can be used to input numerical values to a specified address. It can also be set to read values from an address and display them on the plugin.

### 6.7.2. Component Structure

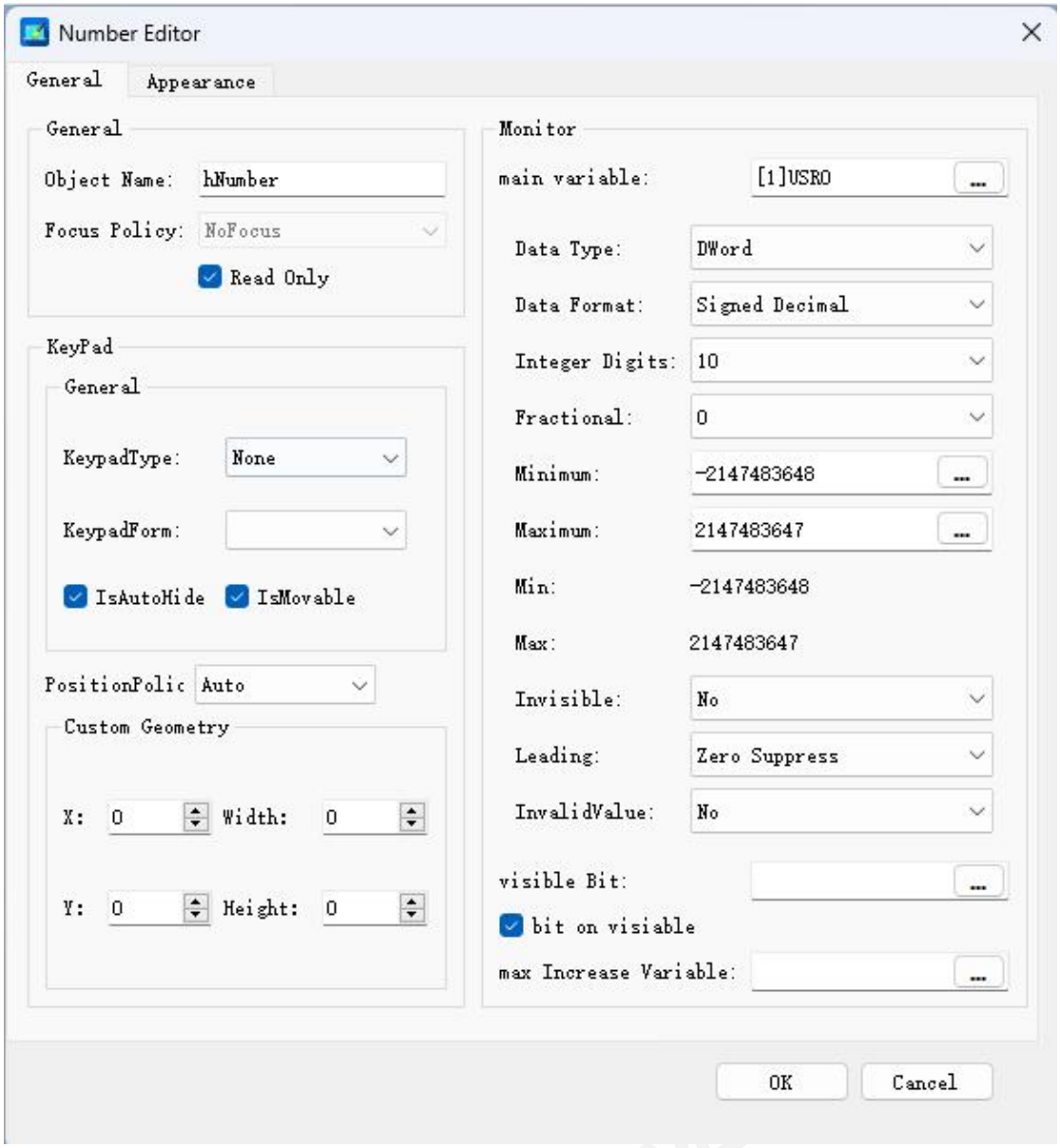



The numeric component is simple, consisting of a border and a display area.

### 6.7.3. Property Settings Popup

In the HMI software editing interface, double-click the component with the left mouse button to bring up the property settings interface. The property settings popup includes two tabs: "General Properties" and "Appearance & Shape".

## 6.7.3.1. General Properties




**1. General Properties**

1) Object Name

Each object can be identified and accessed by setting its Object Name, which is a string used to reference and manipulate objects in the code.

2) Focus Policy

Determines how the widget acquires focus and handles focus events.

3) Read-only

Checking this option sets the component to read-only mode, preventing users from editing the value.

**2. Virtual keyboard.**

When using touch input, you can use the virtual keyboard.

**3. Monitored Variables**

1) Linked Variables

Set the variables associated with the plugin. You can input variables using the variable editing popup.

2) Virtual keyboard

| unit | length |
|---|---|
| Word | 字（16Bit） |
| DWord | 双字(32Bit) |

3) Numeric types

| numeric types | description |
|---|---|
| Floating | Floating point |
| Signed Decimal | Signed decimal |
| Unsigned Decimal | Unsigned decimal |
| Signed BCD | Binary-coded decimal, Signed |
| BCD | Binary-coded decimal, Unsigned |
| Hexadecimal | Hexadecimal |
| Binary | Binary |

Below are the soft keyboard types corresponding to different numerical data types.

| Numeric types | Types of Software Keyboards |
|---|---|
| Floating<br>Signed Decimal<br>Unsigned Decimal | Decimal Keyboard |

| | |
|---|---|
| Signed BCD<br>BCD |  |
| Hexadecimal | Hexadecimal Keyboard<br>  |
| Binary | Binary Keyboard<br>  |

4) Integer Digits

Set the number of integer digits.

5) Decimal Places

When using floating-point data types, you can set the number of decimal places.

6) Minimum/Maximum Value

Set the valid range for numerical input.

| Numerical input | | |
|---|---|---|
| Word | BCD | 0~9999 |
| | Signed BCD | -999~9999 |
| | Signed Decimal | -32768~32767 |
| | Unsigned Decimal | 0~65535 |
| | Hex | 0-0xFFFF |
| | Binary | 0-0xFFFF |
| DWord | BCD | 0-99999999 |
| | Signed BCD | -9999999~99999999 |
| | Signed Decimal | -2147483648~2147483647 |
| | Unsigned Decimal | 0~4294697295 |
| | Hex | 0~0xFFFFFFFF |
| | Binary | 0~0xFFFFFFFF |

7) Display as Asterisk

Show the value as an asterisk (*).


8) Leading padding

| Padding Options | Specific |
|---|---|
| No padding | No padding |
| Padded with zeros | Padded with '0' characters   |
| Padded with spaces | Padded with spaces |

| | |
|---|---|
| |  |

9) Enable Invalid Value

When this option is enabled, if an empty value is entered, the invalid value "0xBBBBBB" will be written to the variable.

10) Visibility Bit

Link a variable to set whether the widget is visible.

11) Visible When Bit is ON

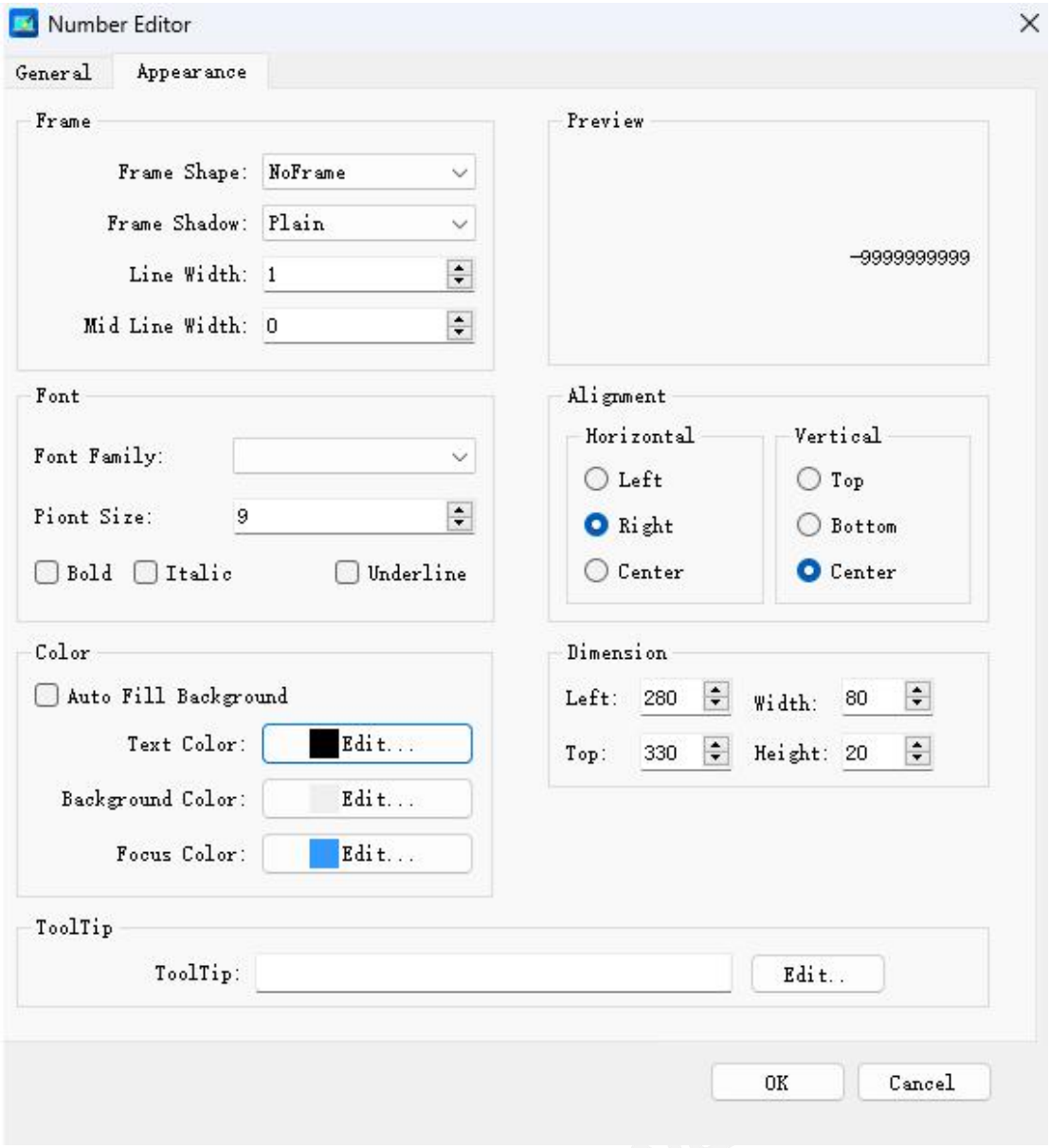
| State | Details |
|---|---|
| checked | When the "Visible Bit" variable is 1, the plugin is visible. Otherwise, it's hidden. |
| unchecked | When the "Visible Bit" variable is 0, the plugin is visible. Otherwise, it's hidden. |

12) Maximum Step Size Variable

The maximum step size is associated with a variable.

| Input Method | Result |
|---|---|
| Soft Keyboard | When the numeric value changes by more than the maximum step size, the input for that instance is invalid. |
| Input Block | When the numeric value changes by more than the maximum step size, "invalid inputs! (abs(input value - current value) > step size)" is displayed in the "Input Block Plugin," and the input for that instance is invalid. |

## 6.7.3.2. Appearance Shape




### 1. Frame

Set the appearance shape of the frame.

### 2. Preview

Preview the effect of the appearance shape settings.

### 3. Font

Set the font for displaying text.

**4. Alignment**

Set the text alignment strategy.

**5. Color**

1) Auto Fill Background Color

When the auto-fill background color function is enabled, the plugin will automatically fill the background color based on the current color scheme.

2) Text Color

Set the text color.


3) Background Color

Set the background color.


(Note: The background color is effective when the "Auto Fill Background Color" option is enabled.)

4) Focus Color

Set the background color of the plugin when it has focus.

**6. Tool Tips**

Used in conjunction with the Label plugin, set the content displayed by the Label component's "Tool Tip" feature when the plugin is focused or when the cursor hovers over the plugin.

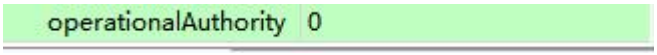
## 6.7.4. Property Editor

The Property Editor provides a convenient way to set properties without needing to double-click to open the property setting pop-up window. Most of the property settings in the Property Editor function the same as those in the property setting pop-up window. Here, we will only introduce functions not covered in the property setting pop-up window.

### 1. Use Input Block

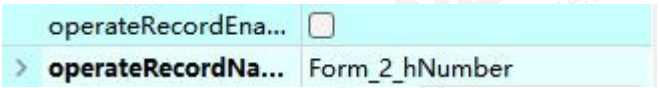

When "Use Input Block" is checked, the input content is displayed in the "Input Block" plugin. After clicking the "Enter" button, the content is then written into the numerical plugin.


### 2. Operation Permissions




Set the operation permissions for the component. If the permissions are insufficient, the numerical plugin will not be editable.

### 3. Operation Records




Enable the operation record function for the component.

When "Linked Variable," "Visible Bit," or "Maximum Step Variable" changes, these changes are recorded in the operation records.

| | 时间 | 记录者 | 操作 | 详情 |
|---|---|---|---|---|
| 1 | 2023-08-29 16:58:30:738 | Form_hNumber_2 | 变数改变 | [0]USR0,13->90 |
| 2 | 2023-08-29 16:56:50:382 | System | 0.8.1.254 | 初始化过程中出现错误或者超时，初始化停止 |
| 3 | 2023-08-29 16:56:50:333 | System | 0.8.1.3 | 获得从站数量过程中 |
| 4 | 2023-08-29 16:56:50:301 | System | 0.8.1.2 | 设定通讯周期过程中 |
| 5 | 2023-08-29 16:56:50:268 | System | 0.8.1.1 | 创建FRAME过程中 |
| 6 | 2023-08-29 16:56:50:199 | System | -1.7.7 | 保存公共变数 COM（MCM）完成 |
| 7 | 2023-08-29 16:56:50:198 | System | 2.6.1 | 通道2 控制器错误 |
| 8 | 2023-08-29 16:56:50:186 | System | 0.2.1 | 公共通道 Reset |
| 9 | 2023-08-29 16:56:50:142 | System | -1.7.2 | 保存-1 MCM变数完成 |
| 10 | 2023-08-29 16:56:50:114 | System | -1.7.2 | 保存-1 MCM变数完成 |
| 11 | 2023-08-29 16:56:50:047 | System | 0.8.1.0 | 开始初始化 |
| 12 | 2023-08-29 16:56:50:046 | System | 0.3.0 | 切换至画面0 |
| 13 | 2023-08-29 16:56:50:025 | System | 1.2.1 | 通道1 Reset |
| 14 | 2023-08-29 16:56:50:022 | System | 2.2.1 | 通道2 Reset |

**4. =Operation Record Name**

| | |
|---|---|
| operateRecordEna... | ☐ |
| **operateRecordNa...** | Form_2_hNumber |

Set the display name of the component in the operation record table.

## 6.7.5. Action Editor

Action Editor

| Event | Macro |
|---|---|
| beforValueChanged | |
| editingFinished | |

**1. Before Value Change**

This event is triggered before the value changes during the editing process. (Note: Changes to the "associated variable" will not trigger this event)

**2. Editing Completed**

This event is triggered when the value editing is completed during the editing process. (Note: Changes to the "associated variable" will not trigger this event)

### 6.7.6. Script Macros

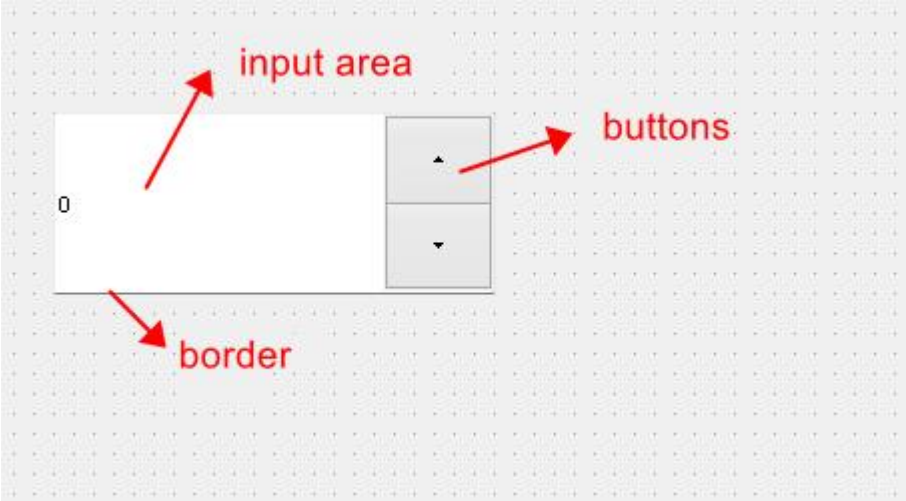
Details can be found in the "Number_Designer_CN.xml" file within the macroWizard folder.

## 6.8. Integer and Decimal Input Boxes

### 6.8.1. Function Introduction

1. The integer input box provides an integer selector with increment and decrement buttons. Users can adjust the integer value by clicking the buttons or manually entering the value.
2. The decimal input box provides a decimal selector with increment and decrement buttons. Users can adjust the decimal value by clicking the buttons or manually entering the value.

### 6.8.2. Component Structure

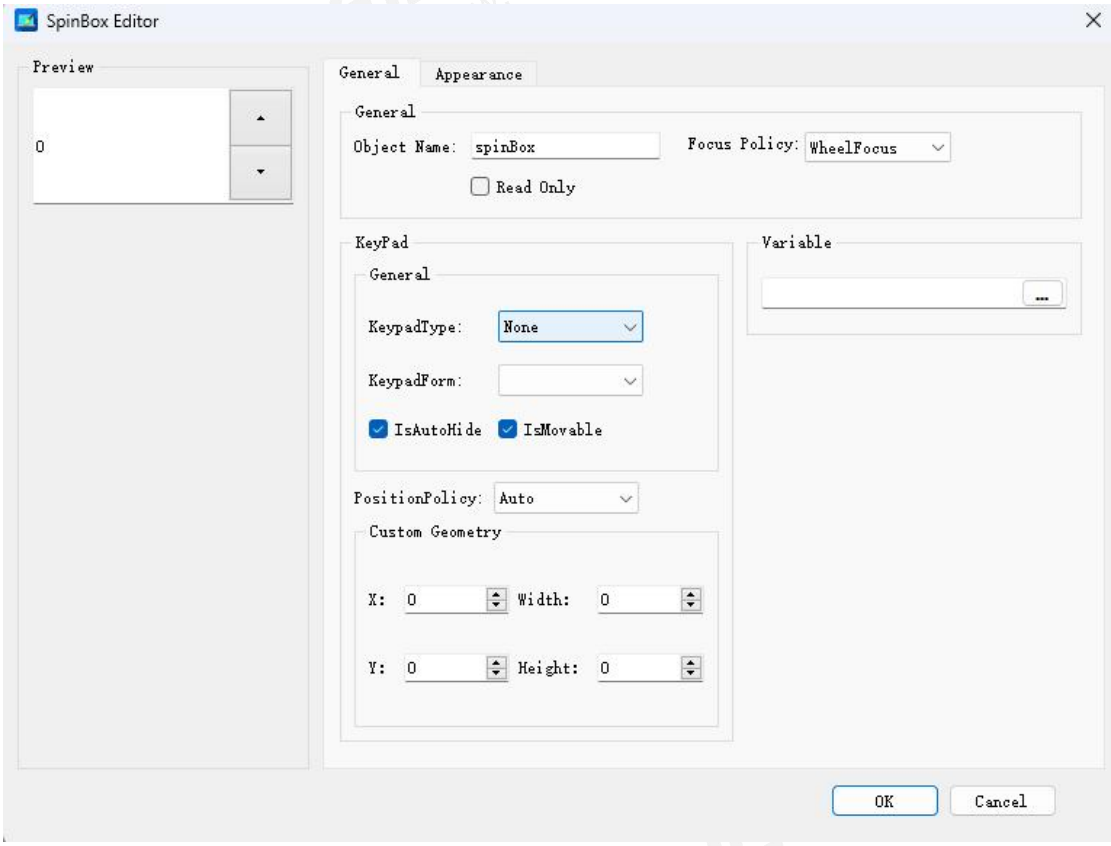



The integer and decimal input boxes have a simple structure, consisting of a border, input area, and buttons.

## 6.8.3. Attribute Settings Popup

In the HMI software editing interface, double-click the component with the left mouse button to bring up the attribute settings interface. The attribute settings popup contains a "General Properties" page.

### 6.8.3.1. General Properties




**1. General**

1) Object Name

Each object can be identified and accessed by setting its object name (Object Name). The object name is a string used to reference and manipulate the object in code.

2) Focus Policy

The Focus Policy determines how the widget acquires focus and handles focus events.

3) Read-Only

Check this option to set the component to read-only mode, preventing users from editing integer values.

### 2. Virtual Keyboard

When inputting using touch, a virtual keyboard can be used. (refer to Virtual Keyboard explanation)

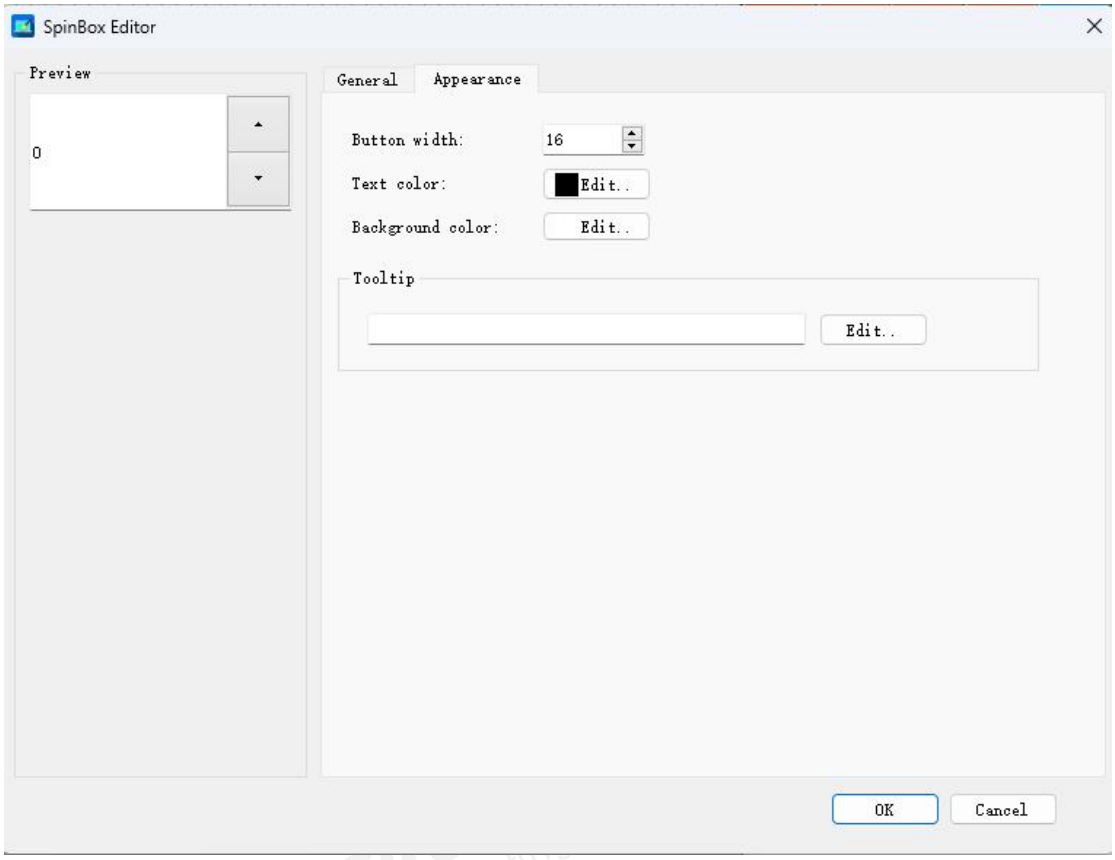
### 3. Variable

Set the variable associated with the component. You can use the variable editing popup to input the variable. (refer to Variable Editing Popup introduction)

After associating the variable, the displayed value is read from the corresponding address.

After modifying the value, the variable's value will also change accordingly.

## 6.8.3.2. Appearance




### 1. Button Width

Set the width of the increment and decrement buttons.


### 2. Text Color

Set the text color.


### 3. Background Color

Set the background color.


**4. Tooltip**

Set the content displayed by the "Tooltip" feature of the label component when it is focused or when the cursor hovers over the plugin.

## 6.8.4. Property Editor

The property editor offers a convenient way to set properties without needing to double-click to open the property settings dialog. Most of the property settings in the property editor have functions consistent with those in the property settings dialog. Here, we only introduce functions not covered in the property settings dialog.

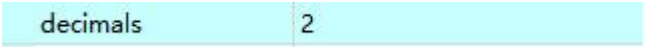
**1. Properties Unique to Integer Input Box**

1) Display Integer Base


The integer base defines the numerical base for displaying integers. By default, the value of displayIntegerBase is 10, meaning integers are displayed in decimal form. Choosing an integer base of 2, 8, 10, or 16 corresponds to binary, octal, decimal, and hexadecimal respectively.

**2. Properties Unique to Floating Point Input Box**

Decimal Places




Used to set the number of decimal places for displaying and inputting floating-point values.

### 3. Properties Shared by Integer and Floating Point Input Boxes

2) Enable Wrap-around

Sets whether the value should wrap around when reaching the maximum or minimum value (using the mouse wheel or clicking buttons).

Example: Minimum value 0, maximum value 100. When the value reaches 100 and continues to increase, the value becomes 0.

3) Border

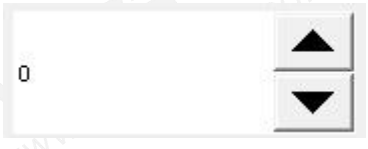
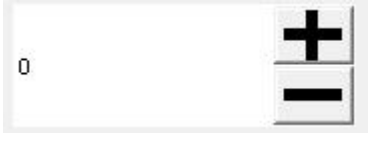
Sets whether the component uses a border.

With Border:


Without Border:


4) Button Symbols

buttonSymbols UpDownArrows

| Option | Style |
|---|---|
| UpDownArrows |   |
| PlusMinus |   |

| | |
|---|---|
| NoButtons | 0 |

5) Special Value Text

specialValueText

Sets the text displayed when the value is less than or equal to the minimum value.

Example: Set the special value text to "Invalid!" and the minimum value to 0. When the value is less than or equal to 0, the special value text is displayed.


6) Speed up

accelerated

By default, the increment and decrement speed is fixed. No matter how long the user holds down the arrow key or scrolls the mouse wheel, the value changes by the same amount each time. When acceleration mode is enabled, if the user continuously holds down the arrow key or scrolls the mouse wheel, the value will change faster, accelerating the increment or decrement process. In most cases, acceleration mode can enhance user experience.

7) Thousands Separator


The thousands separator is a symbol used to group large numbers in a readable way, typically a comma or space. For example, the number 1000000 is displayed as "1,000,000".

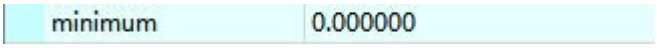
8) Suffix


A suffix string is additional text added at the end of the displayed value. It can be used to denote units, symbols, or any other relevant information.

9) Prefix

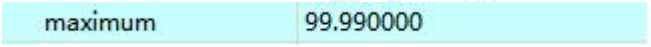

A prefix string is additional text added at the beginning of the displayed value. It can be used to denote units, symbols, or any other relevant information.

10) Minimum Value




The minimum value defines the smallest allowable value. The user cannot set the value to be less than the minimum value either by direct input or by using the increment/decrement buttons.

11) Maximum Value




The maximum value defines the largest allowable value. The user cannot set the value to be greater than the maximum value either by direct input or by using the increment/decrement buttons.

12) Single Step


The single step increment defines the amount by which the value increases or decreases when the user clicks the up or down arrow buttons or uses the up or down arrow keys on the keyboard.

13) Step Type

| stepType | DefaultStepType |
|---|---|

| Option | Description |
|---|---|
| DefaultStepType | Uses single-step increment (singleStep). |
| AdaptiveDecimalStepType | Adjusts the step size based on the magnitude of the current value during input. If the current value is 1000, the step size will adjust to 100. If the current value is 100, the step size will adjust to 10. This ensures that users can more precisely adjust the value and that the step size matches the current value's magnitude. |

14) Value

| value | 99 |
|---|---|

Sets the current value.

15) Button Width

| buttonWidth | 16 |
|---|---|

Adjusts the width of the buttons.

16) Operation Permissions

| operationalAuthority | 0 |
|---|---|

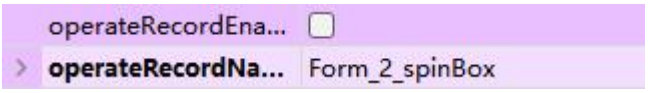
Sets the operational permissions for the component. If permissions are insufficient, the numerical plugin cannot be edited.

17) Enable

| enabled | ☑ |
|---|---|

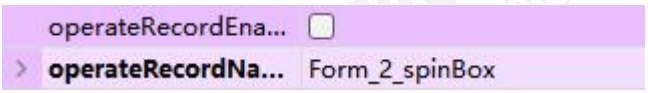
Enables or disables the component. When disabled, users cannot interact with the control and cannot change its value.

18) Operation Records



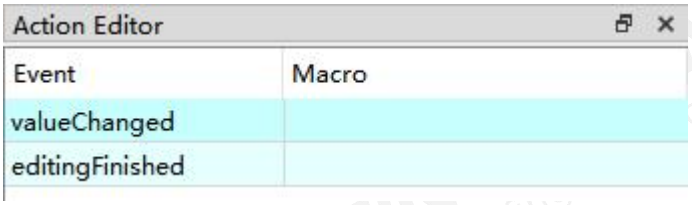

Enables operation recording functionality for the component. Changes in associated variable addresses and values are recorded in the operation log.

19) Operation Record Name




Sets the display name of the component in the operation record table.

## 6.8.5. Action Editor




**1. Value Changed**

Triggered when the value changes

**2. Editing Completed**

Issued when editing is completed. This event is triggered when the user finishes editing, such as pressing the Enter key or leaving the component.

## 6.8.6. Script Macro

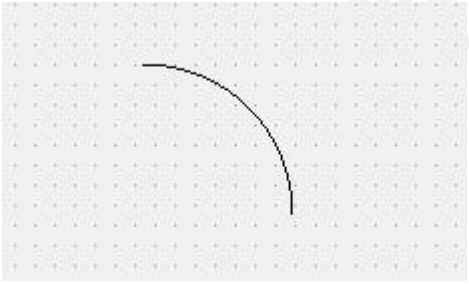
Detailed configurations can be found in the "SpinBox_Designer_CN.xml" and "DoubleSpinBox_Designer_CN.xml" files located in the macroWizard folder.

# 6.9. Arc

## 6.9.1. Functionality Overview

Draws a segment of an arc.

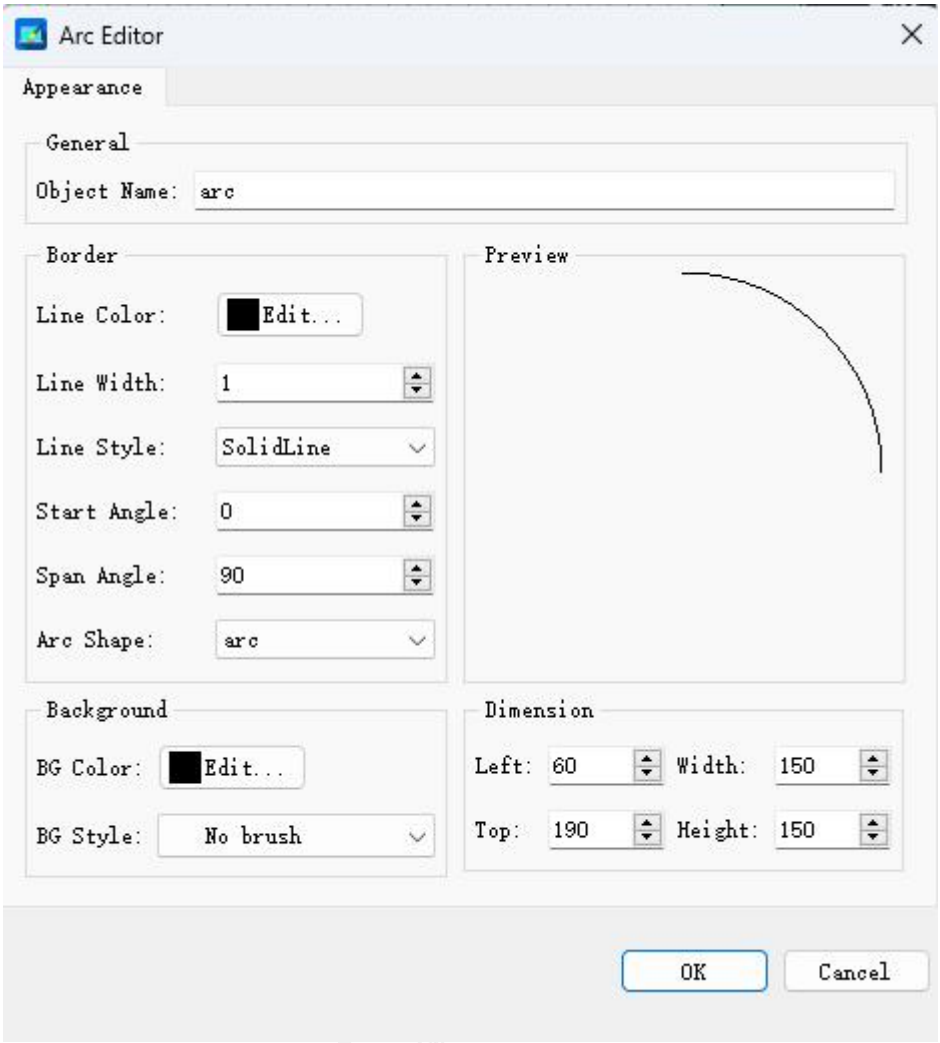
## 6.9.2. Component Structure



The arc component consists solely of an arc.

## 6.9.3. Property Settings Window

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings window. The property settings window includes the "Appearance" page.




## 1. General Properties

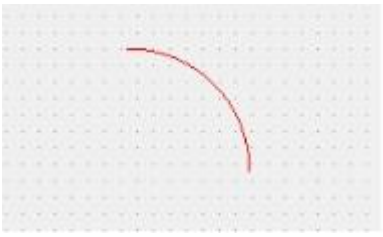
1) Object Name

Each object can be identified and accessed by setting its object name. The object name is a string used to reference and manipulate objects in code.

## 2. Border

2) Line Color

Sets the line color.



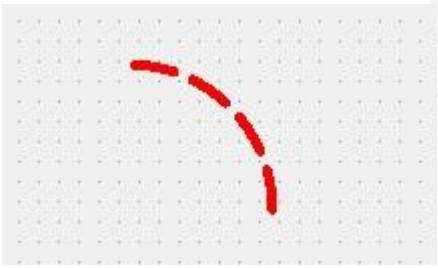
3) Line Width

Sets the line width.

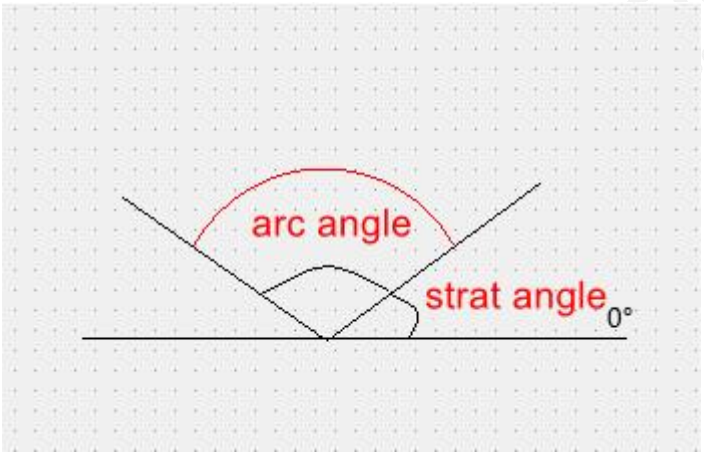
### 3. Line Type

Sets the line style.

The arc style when set to a dashed line type is as follows:

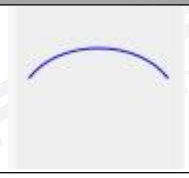
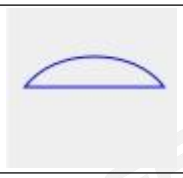


### 4. Start Angle, End Angle (Arc Angle)



Positive values indicate counterclockwise direction, while negative values indicate clockwise direction. Zero degrees is at the 3 o'clock position.

### 5. Arc Shape

| Options | Description |
|---|---|
| Arc |  |
| Pie |  |
| Chord |  |

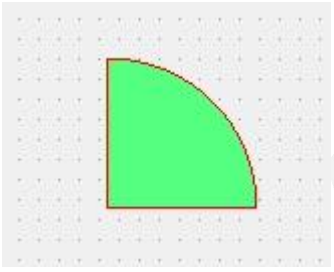
### 6. Preview

Preview the effect of appearance settings.

### 7. Background

1) Background Color

Set the fill color for the enclosed shape.



(Note: When the arc shape is set to Pie  or Chord  , the background setting

applies if the brush fill type is not "No Brush".)

2) Fill Type

Set the style of the brush.

(Note: When the arc shape is set to Pie  or Chord  , the fill type of the brush applies.)

**8. Geometry**

This property controls the position and size of the component on the screen.

## 6.9.4. Property Editor

The property editor provides a convenient way to set properties without the need to double-click to open the property setting dialog. The functionality of the properties in the arc component's property editor is consistent with that of the property setting dialog. Refer to the property setting dialog for details on each function.

## 6.9.5. Action Editor

The arc component does not trigger any events.

## 6.9.6. Script Macros

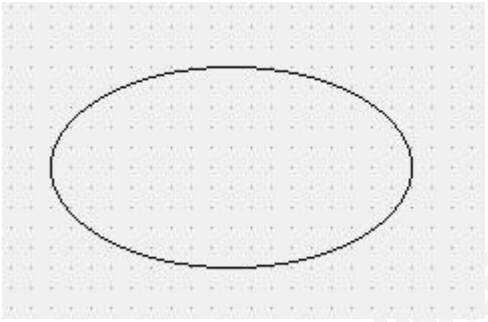
Specific details can be found in the "Arc_Designer_CN.xml" file in the macroWizard folder.

## 6.10. Ellipse

## 6.10.1. Function Introduction

Draws an elliptical arc.

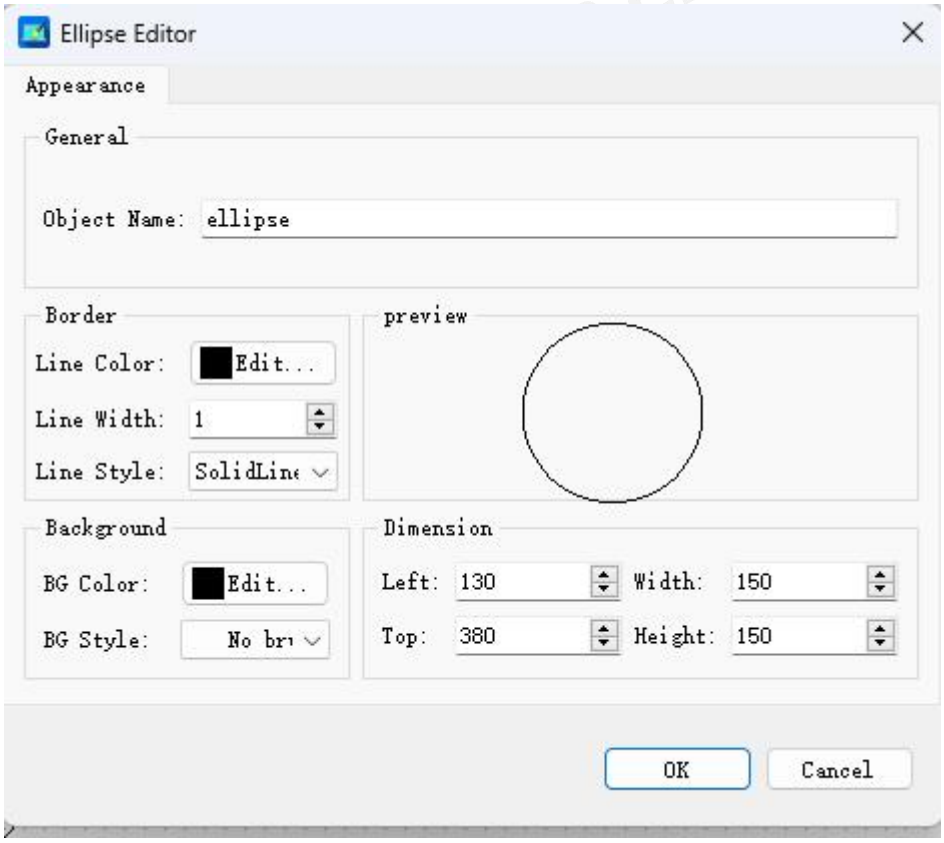
## 6.10.2. Component Structure



The ellipse component consists solely of an ellipse.

## 6.10.3. Property Settings Window

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings window. The property settings window includes the "Appearance" page.




### 1. General Properties

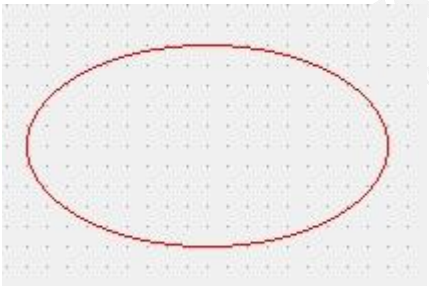
1) Object Name

Each object can be identified and accessed by setting its object name. The object name is a string used to reference and manipulate objects in the code.

## 2. Border

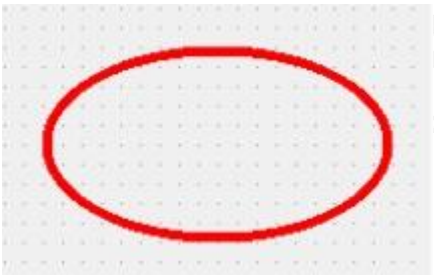
2) Line Color

Sets the color of the line.



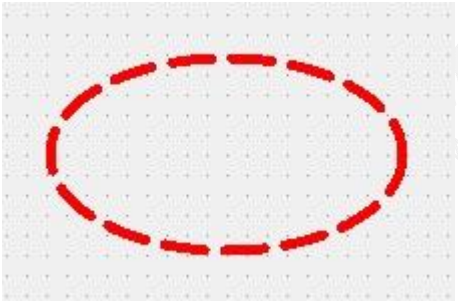
3) Line Width

Set the width of the line.



## 3. Line Type

Set the style of the line. When set to a dashed line type, the ellipse style looks like this:



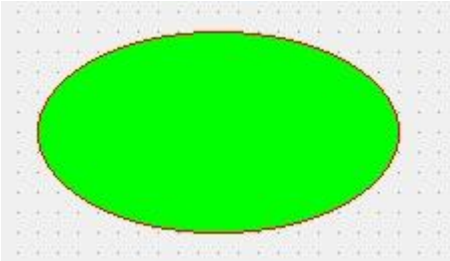
**4. Preview**

Preview the effects of the appearance settings.

**5. Background**

4) Background Color

Set the fill color of the ellipse component.



(Note: Background color setting applies when the fill type of the brush is not "No Brush")

5) Fill Type

Set the style of the brush.

**6. Geometry Size**

This property controls the position and size of the component on the screen.

## 6.10.4. Property Editor

The property editor provides a convenient way to set properties without needing to double-click to open a property settings dialog box. The functionality of the ellipse component's property editor is consistent with the property settings dialog box. You can refer to the property settings dialog box to understand each function.

## 6.10.5. Action Editor

The ellipse component does not trigger any events.

### 6.10.6. Script Macros

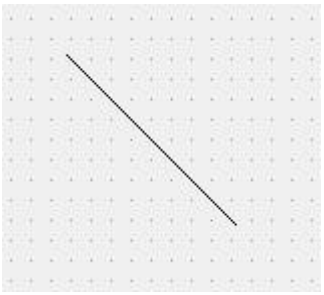
For specific details, refer to the "Ellipse_Designer_CN.xml" file in the macroWizard folder.

## 6.11. Straight Line

### 6.11.1. Function Introduction

Draws a straight line.

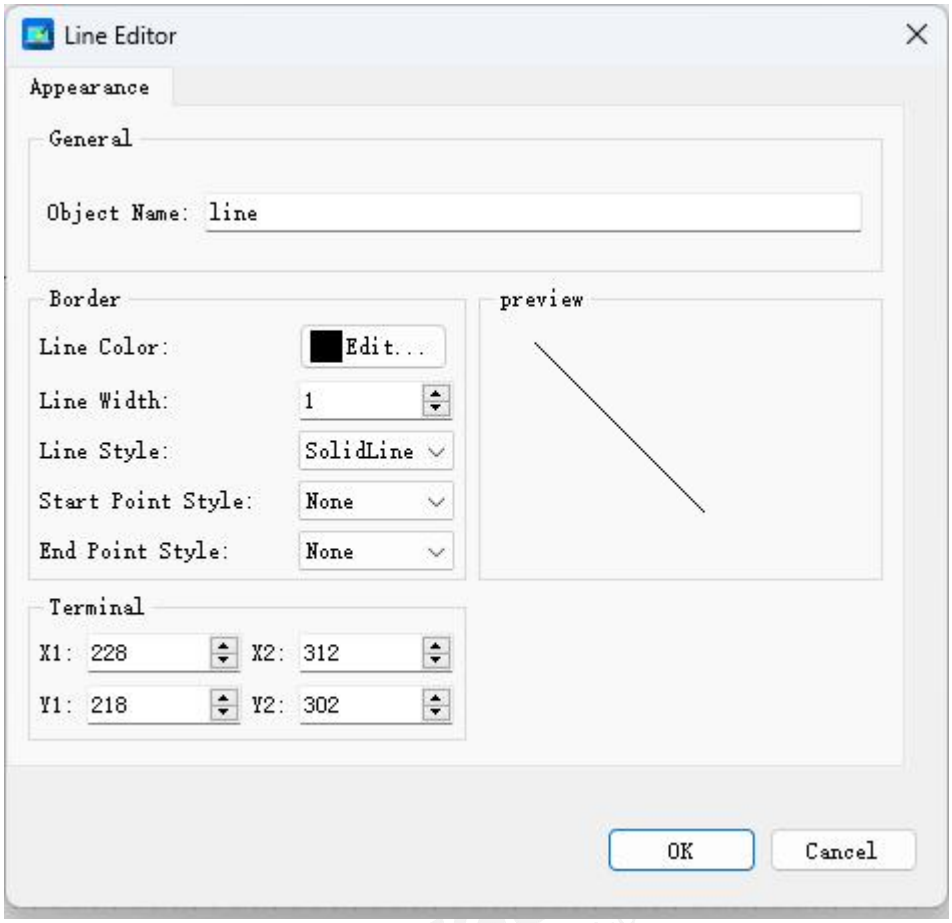
### 6.11.2. Component Structure



The straight line component consists solely of a straight line.

### 6.11.3. Property Settings Dialog Box

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings dialog box. The property settings dialog box includes the "Appearance" page.




## 1. General Properties

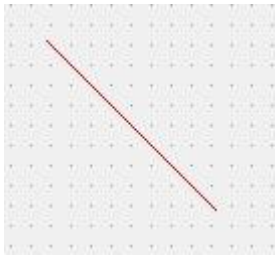
1) Object Name

Each object can be identified and accessed by setting its object name (Object Name). The object name is a string used to reference and manipulate objects in code.

## 2. Border

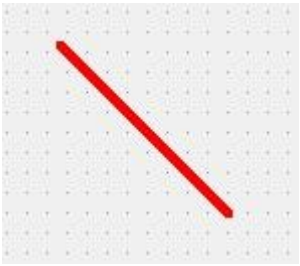
2) Line Color

Sets the color of the line.



3) Line Width

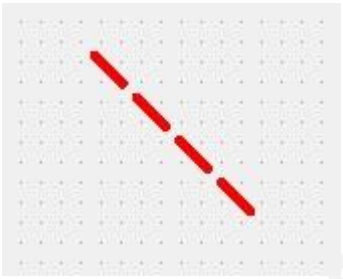
Sets the width of the line.



4) Line Type

Sets the style of the line.

The appearance of the line when set to a dashed line type is as follows:



5) Endpoint style

| Options | Style |
|---|---|
| None |  |
| Arrow |  |

| | |
|---|---|
| Dot |  |

**3. Preview**

Preview the appearance settings effect.

**4. Endpoints**

Set the endpoint coordinates of the straight line.

## 6.11.4. Property Editor

The property editor provides a convenient way to set attributes without the need to double-click and open the attribute settings dialog. The functionality of the ellipse component's property editor is identical to that of the attribute settings dialog. Please refer to the attribute settings dialog for details on each function.

## 6.11.5. Action Editor

There are no event triggers for the Line component.

## 6.11.6. Script Macros

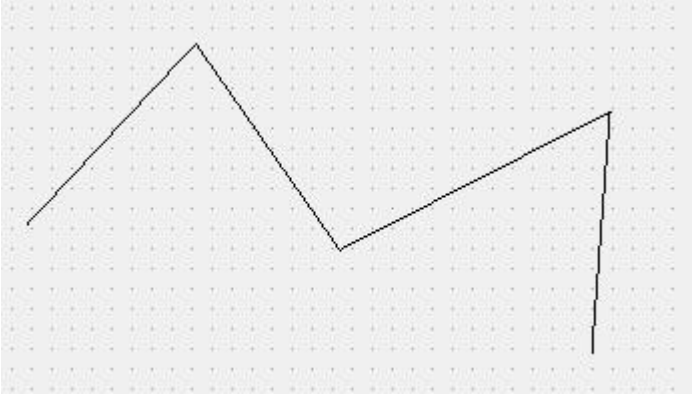
For details, refer to the 'Line2_Designer_CN.xml' file in the macroWizard folder.

## 6.12. Polygon

## 6.12.1. Function Introduction

Draws a polygon.

## 6.12.2. Component Structure

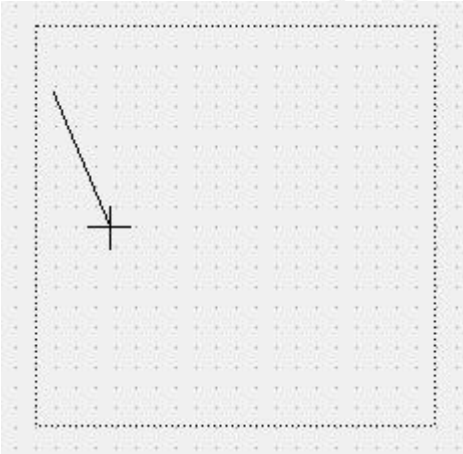


The Polygon component consists solely of multiple line segments.

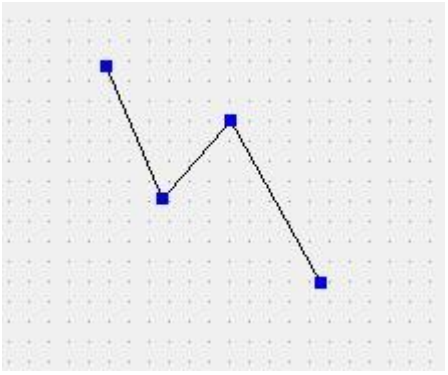
## 6.12.3. Editing Method

In the HMI editing software, use the following method to edit polygons:

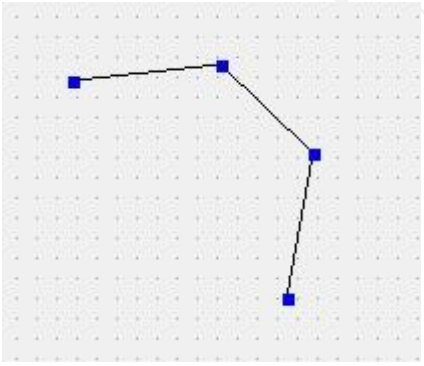
1) Within the dashed rectangle area, click with the left mouse button to control the vertices of the polygon. Double-click with the left mouse button to complete the drawing of the polygon.



2) After completing the drawing, the dashed rectangle disappears. When the polygon is selected, its vertices are displayed in blue rectangles.

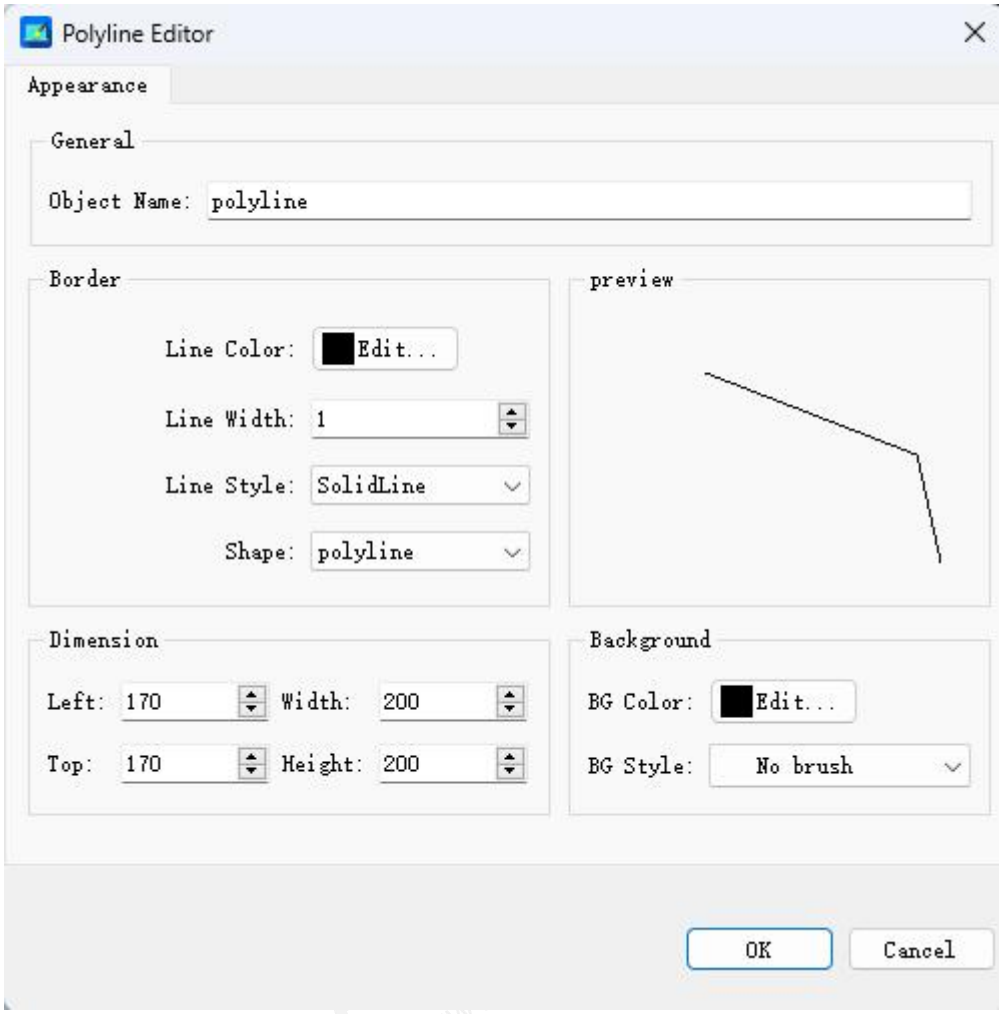


3) You can adjust the polygon by dragging its vertices.



## 6.12.4. Property Settings Dialog

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings dialog. The dialog includes the 'Appearance' page.




## 1. General Properties

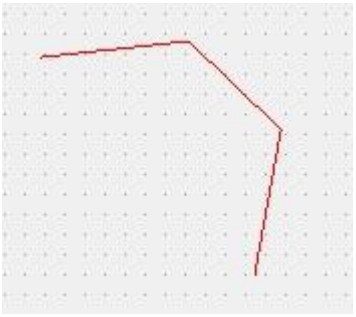
1) Object Name

Each object can be identified and accessed by setting an Object Name. The Object Name is a string used to reference and manipulate objects in code.

## 2. Border

2) Line Color

Set the color of the line.



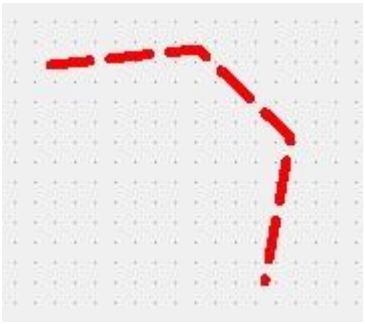
3) Line Width

Set the width of the line.

4) Line Type

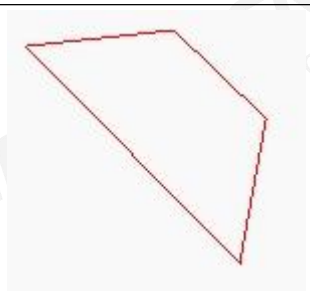
Set the line style.

The polygon's appearance with a dashed line type is as follows:



5) Shape

| Options | Style |
| --- | --- |

| | |
|---|---|
| Polyline |  |
| Polygon |  |

### 3. Preview

Preview the appearance settings.

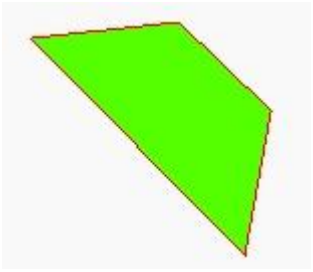
### 4. Geometric Dimensions

This property controls the component's position and size on the screen.

### 5. Background Color

6) Background Color

Background Color Set the fill color of the polygon component.



(Note: When 'Shape' is set to 'Polygon' and the fill type of the brush is not 'No Brush', the background color setting takes effect.)

7) Fill Type

Set the style of the brush.

### 6.12.5. Property Editor

The property editor provides a convenient way to set properties without the need to double-click to open the property settings dialog. The property editor for the polygon component offers the same functionalities as the property settings dialog. Refer to the property settings dialog for details on each function.

### 6.12.6. Action Editor

The polygon component does not trigger any events.

### 6.12.7. Script Macros

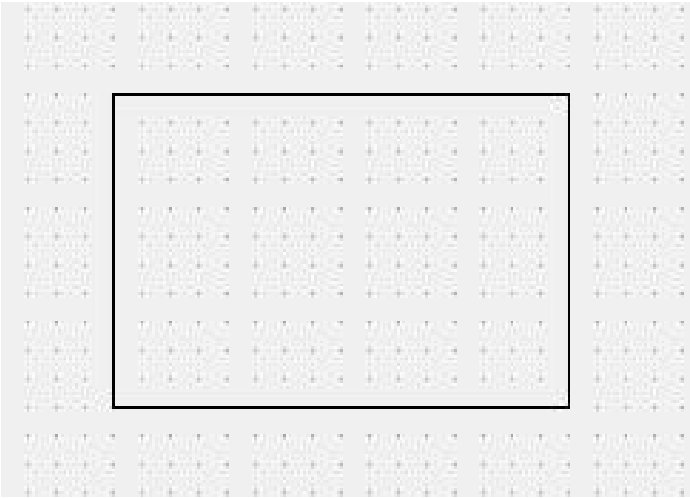
For specific details, refer to the 'Polyline_Designer_CN.xml' file in the macroWizard folder.

## 6.13. Rectangle

### 6.13.1. Functionality Overview

Draws a rectangle.

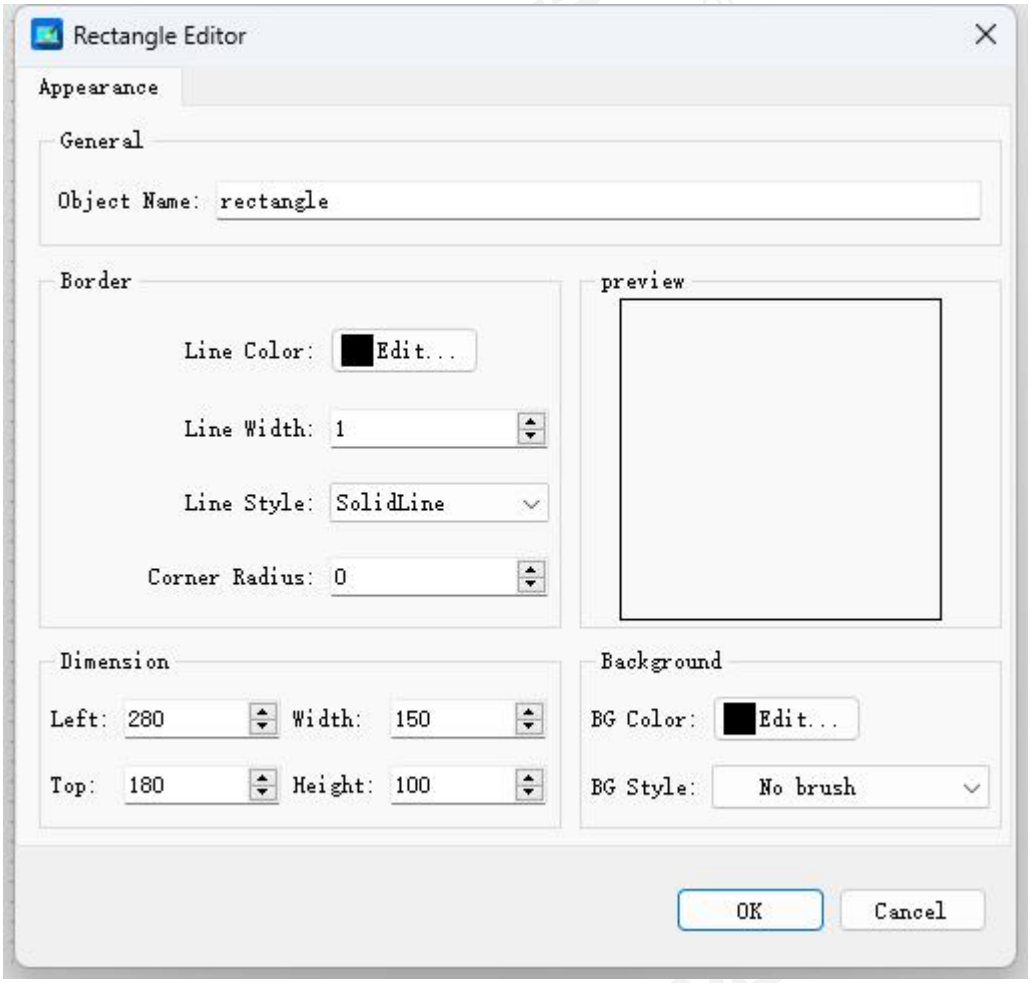
### 6.13.2. Component Structure



The rectangle component consists solely of a rectangular frame.

## 6.13.3. Property Settings Dialog

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings dialog, which includes the 'Appearance' page.




### 1. General Properties

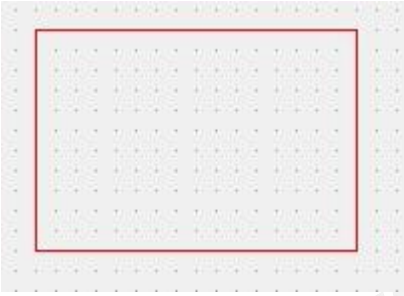
1) Object Name

Each object can be identified and accessed by setting an Object Name. This name is a string used to reference and manipulate objects in code.

### 2. Border

2) Line Color

Sets the color of the border line.

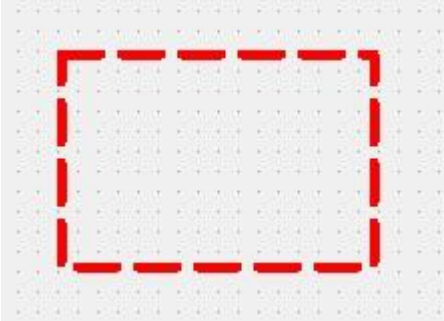


3) Line Width

Adjusts the width of the border line.

4) Line Type

Specifies the style of the line. When set to a dashed line type, the polygon style appears as follows:



5) Corner Radius

Sets the corner radius of the rectangle.

### 3. Preview

Displays a preview of the appearance settings.

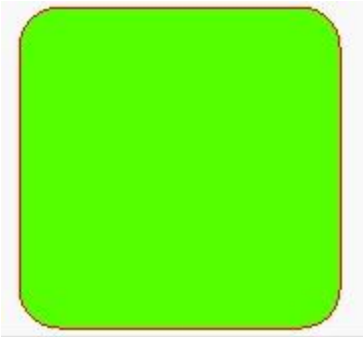
### 4. Geometry Size

Controls the position and size of the component on the screen.

### 5. Background Color

6) Background Color

Sets the fill color of the rectangle component.



(Note: Background color settings take effect when the fill type of the brush is not 'No Brush'.)

7) Fill Type

Set the style of the brush.

### 6.13.4. Property Editor

The property editor provides a convenient way to set properties without the need to double-click to open the property settings dialog. The property editor for the rectangle component functions identically to the property settings dialog. Refer to the property settings dialog for details on each function.

### 6.13.5. Action Editor

The rectangle component does not trigger any events.

### 6.13.6. Script Macros

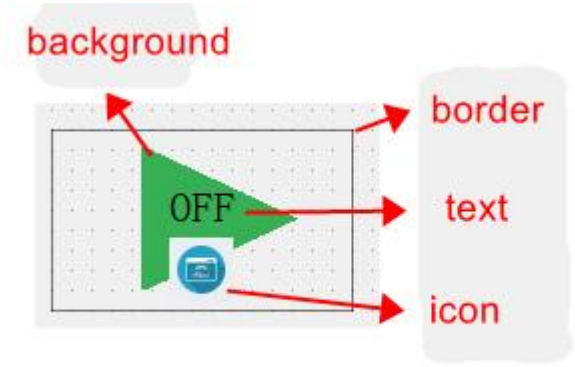
具体可见 macroWizard 文件夹的“Rectangle_Designer_CN.xml”

## 6.14. Toggle Button

### 6.14.1. Function Introduction

1. Read and write system bit variables.
2. Execute script macros by triggering events such as "click," "mouse down," and "mouse up.

## 6.14.2. Component Structure

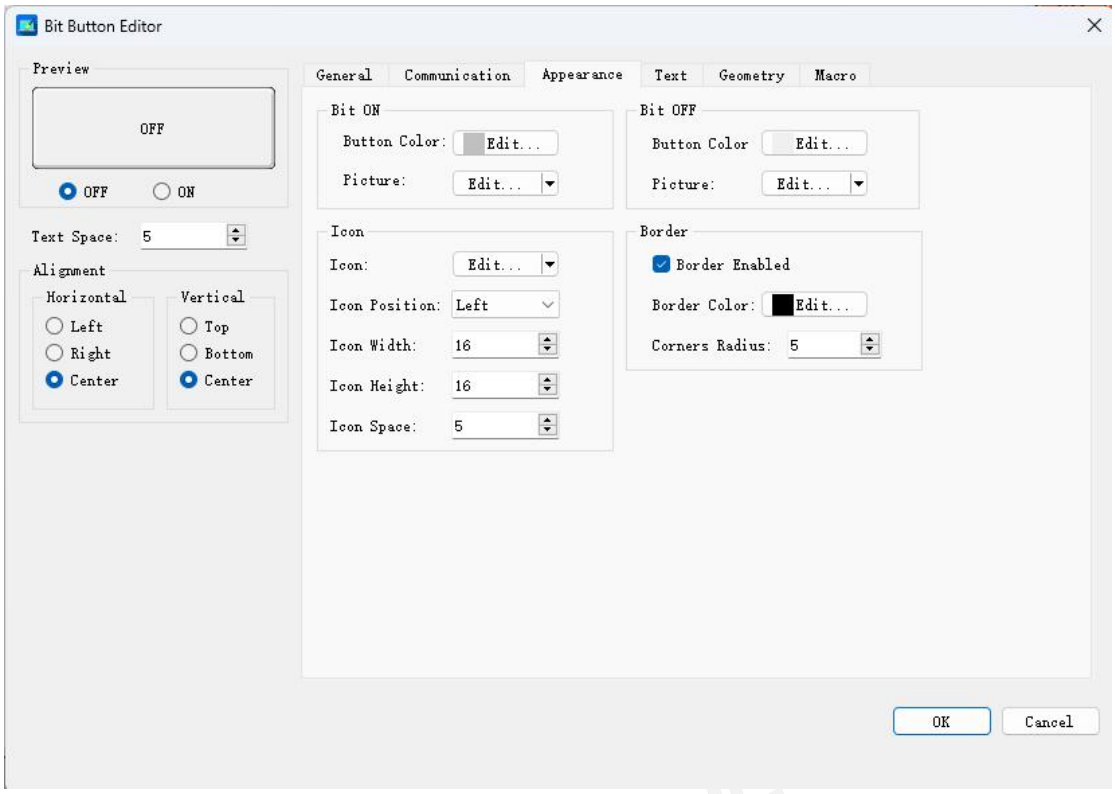



The bit button component consists of "Border," "Text," "Icon," and "Background."

## 6.14.3. Property Settings Popup

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings popup. The property settings popup includes five pages: "General Properties," "System Variable Linking," "Appearance," "Text," "Geometry Size," and "Macro." Additionally, it features settings for "Preview," "Text Spacing," and "Alignment."

[Preview, Text Spacing, Alignment]




### 1. Preview

State OFF: Appearance of the button when it is in the OFF state.

State ON: Appearance of the button when it is in the ON state.

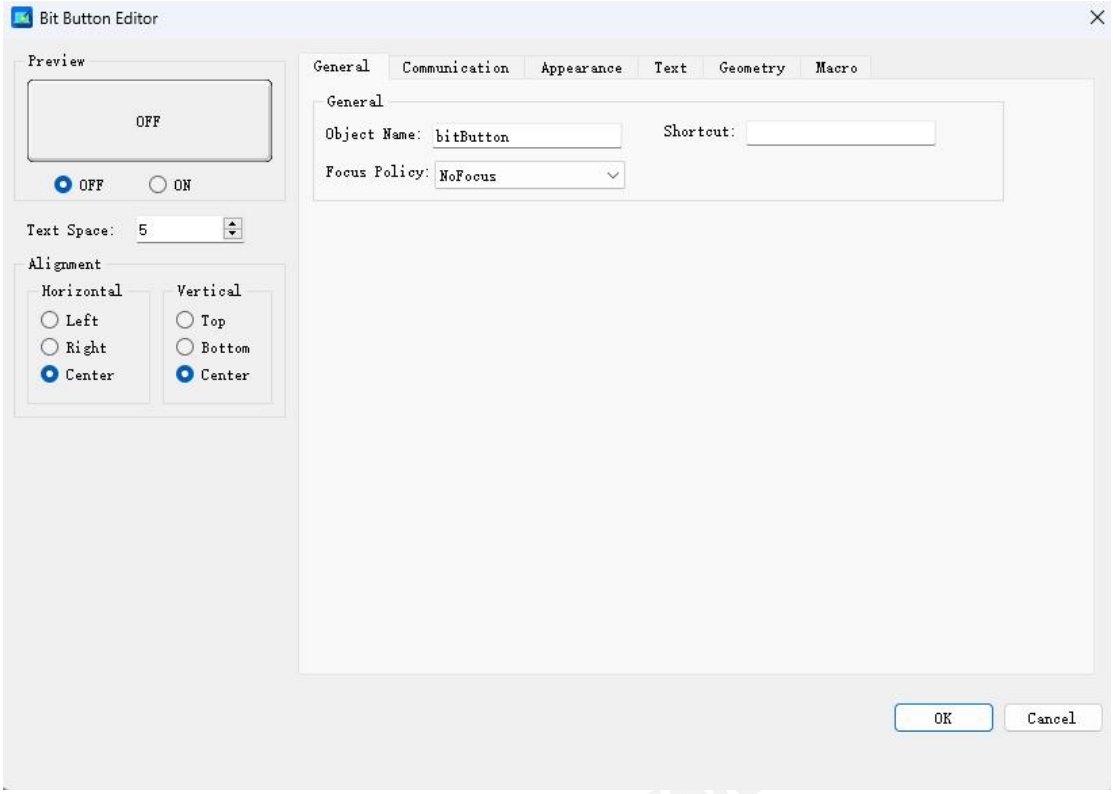
### 2. Text Spacing

Sets the spacing between the button icon and the text.

### 3. Alignment

Sets the text alignment strategy. For details, refer to "Alignment." Note that when an icon is set, the text alignment strategy becomes inactive, and the icon positioning strategy takes effect.

[General Properties]




## 1. General Properties

1) Object Name

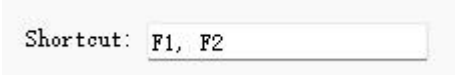
Each object can be identified and accessed by setting an object name (Object Name). The object name is a string used to reference and manipulate objects in code.

2) Focus Policy

Focus Policy determines how the widget acquires focus and handles focus events.

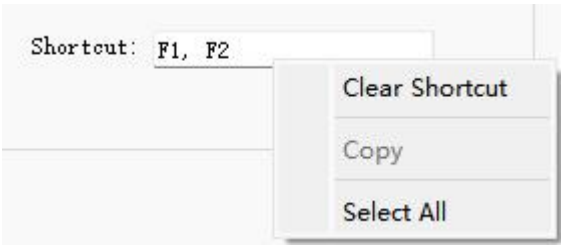
3) Shortcut Keys

Shortcut keys in applications provide a quick way to access and execute specific functions or operations without the need for complex mouse operations. Using shortcut keys can significantly improve user efficiency and operational convenience.

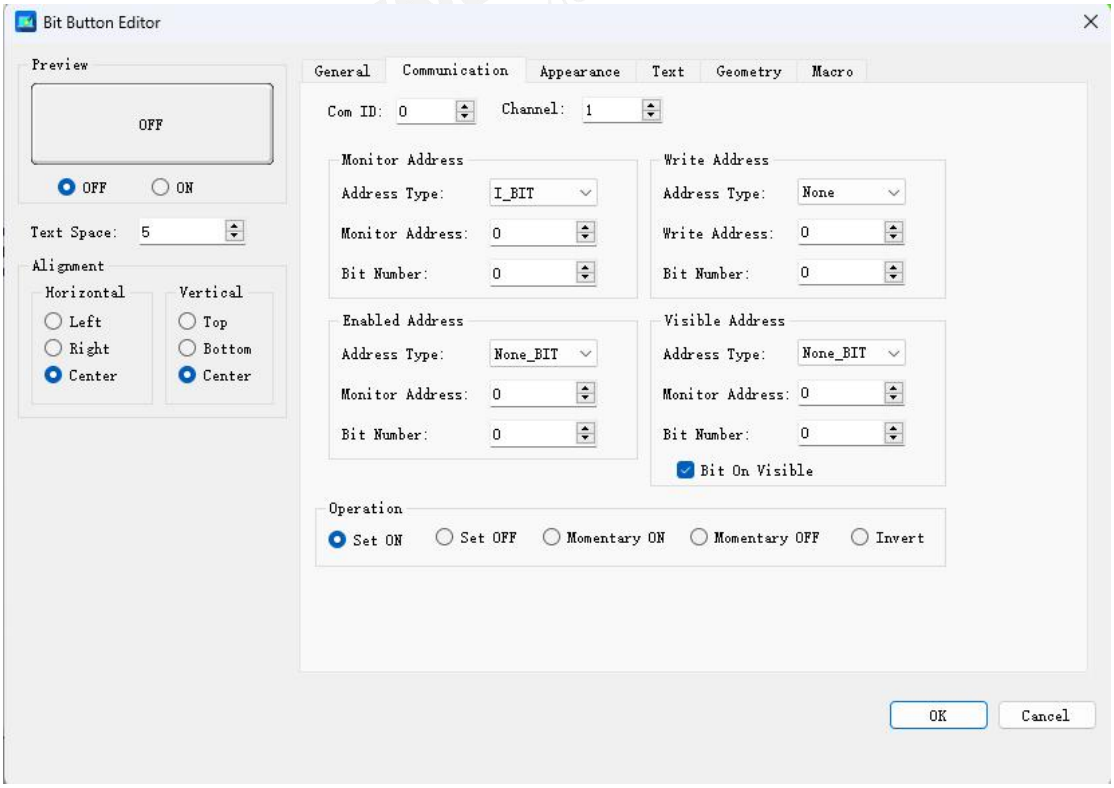



When setting multiple shortcuts, all shortcuts can trigger the button.

Right-clicking on the shortcut input box provides an option to clear the shortcuts.




[System Variable Connection]




There are two types of bit variables:

One type consists of Address Type + Bit Address.

The other type consists of Channel + Address Type + Variable Address + Bit Address.

Specific categories are as follows:

| Bit Variable Types | Address Types |
|---|---|
| Address Type + Bit Address | Example of Bit Variable: S79 (Address Type + Bit Address) |
| Channel + Address Type + | Example of Bit Variable: [1]Usr1.0 ([Channel] Address |

| | |
|---|---|
| Variable Address + Bit Address | Type + Variable Address.Bit Address) |

### 1. Address Type

Set the type of variable address. Options include: None_BIT, I_BIT, O_BIT, C_BIT, S_BIT, A_BIT, Cn_BIT, Tm_BIT, User_BIT, Mcm_BIT, Sys_BIT, Reg_BIT, Bus_BIT, Com_BIT. Note that selecting None_BIT as the address type means no variable association.

### 2. Variable Address

When entering the variable address, pay attention to the currently selected address type.

| Bit Variable Types | Read Variable Address |
|---|---|
| I_BIT, O_BIT, C_BIT, S_BIT, A_BIT, Cn_BIT, Tm_BIT | Fill in Bit Address<br>Example of a bit variable: S79 |
| User_BIT, Mcm_BIT, Sys_BIT, Reg_BIT, Bus_BIT, Com_BIT | Enter Variable Address<br>Example of a bit variable: [1]Usr1.0, which means [Channel] Address type + Variable address.Bit address. |

### 3. Bit Number

The entry of the bit number should take into account the currently selected address type.

| Bit Variable Types | Bit Number |
|---|---|
| I_BIT, O_BIT, C_BIT, S_BIT, A_BIT, Cn_BIT, Tm_BIT | No need to fill in the bit number. |
| User_BIT, Mcm_BIT, Sys_BIT, Reg_BIT, Bus_BIT, Com_BIT | Fill in the bit number for the variable address.<br>Example of a bit variable: [1]Usr1.0, where [Channel] Address Type + Variable Address. Here, "bit address" is synonymous with "bit number." |

### 1. Communication ID

This property is invalid.

### 2. Channel

Sets the channel number for reading and writing variables.

### 3. Read Variable Address

This property can set the state of the bit variable associated with the button. Depending on the switch of the bit variable, it displays as pressed or released.

| Bit Variable State | Button States |
| --- | --- |
| On (when the bit variable is 1) | Pressed   |
| Off (when the bit variable is 0) | Released   |

Changing the button state via a bit variable does not trigger "macro" actions.

### 4. Write Address

This property allows the button state to control a bit variable. Depending on whether the button is in "pressed" or "released" state, it sets the value of the bit variable. The value of the bit variable is related to the "action" settings, details of which can be found under "actions".

### 5. Enable Control

This property determines whether the button is enabled or disabled. When the component is disabled, it automatically ignores user input events such as clicks and keyboard events. Disabled controls typically display a visually distinct style to differentiate them from enabled controls.

| Bit Variable | Enable State |
|---|---|
| On (bit variable is 1) | Enabled  |
| Off (bit variable is 0) |  Note: This is not the "Pressed" state but the "Disabled" state effect. |

## 6. Control Visibility

This attribute sets whether the button is visible. When a control is set to visible, it will appear in the user interface; when it is set to invisible, it will be hidden and the user will not be able to see it.

1) Visible When Bit Is On

Sets the visibility state of the button based on the associated bit variable.

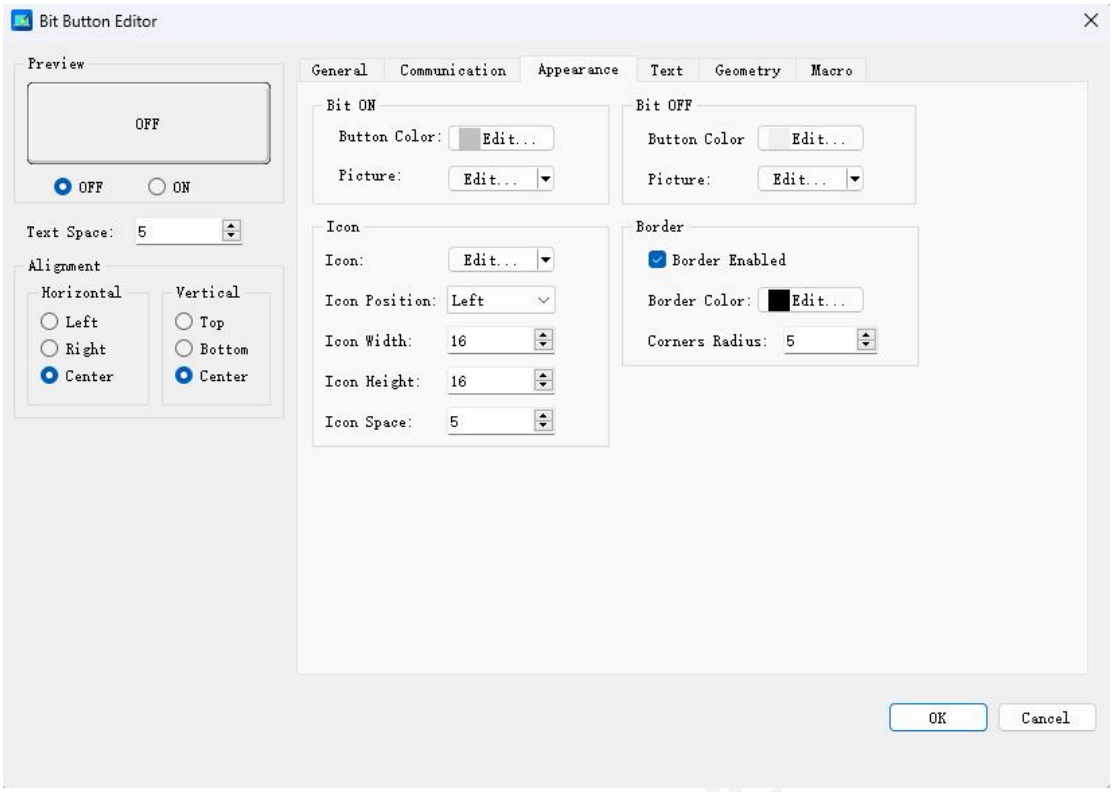
| Visible When Bit Is On | Bit Variable State | Visibility Status |
|---|---|---|
| Checked | On (bit variable is 1) | Visible |
| | Off (bit variable is 0) | Not Visible |
| Unchecked | On (bit variable is 1) | Not Visible |
| | Off (bit variable is 0)） | Visible |

## 7. Action

This property sets how the bit variable associated with the write address (write bit) changes when the button is pressed.

| Actions | Bit Variables |
|---|---|
| Set to 1 | When the button is pressed once, the write bit remains in the "on" state. |
| Set to 0 | When the button is pressed once, the write bit remains in the "off" state. |
| Press to 1 | When the button is pressed, the write bit remains in the "on" state. When the button is released, the write bit remains in the "off" state. |
| Press to 0 | When the button is pressed, the write bit remains in the "off" state. When the button is released, the write bit remains in the "on" state. |
| Toggle | When the button is pressed, if the write bit is in the "on" state, it changes the write bit state to "off"; if the write bit is in the "off" state, it changes the write bit state to "on." |

[Appearance and Shape]



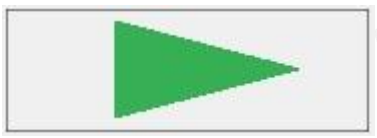

## 1. Bit On State

1) Button Color

Set the background color of the bit button when it is in the "pressed" state.


2) Image

Set the background image of the bit button when it is in the "pressed" state. For details on how to add and remove images, see the "Image Resources" explanation.



The "Image" attribute will override the "Button Color" attribute.

## 2. Bit Off State

3) Button Color

Set the background color of the bit button when it is in the "released" state.

4) Image

Set the background image of the bit button when it is in the "released" state. For details on how to add and remove images, see the "Image Resources" explanation.

The "Image" attribute will override the "Button Color" attribute.

**3. Icon**

5) Icon

Set the icon of the bit button. For details on how to add and remove images, see the "Image Resources" explanation.

6) Icon Position

| Icon Position | Effect |
|---|---|
| Top |  |
| Bottom |  |

| | |
|---|---|
| Left |  |
| Right |  |

7) Icon Width

Adjust the width of the icon.

| Icon Width | Effect |
|---|---|
| 16 |  |
| 32 |  |
| 48 |  |

8) Icon Height

Set the icon height.

| Icon Height | Effect |
|---|---|
| 16 |  |
| 32 |  |

| | |
|---|---|
| 48 |  |

9) Icon margin

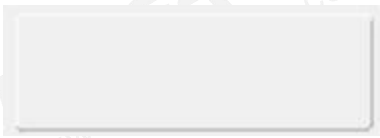
Set the margin between the icon and the button border.

| Icon margin | Effect |
|---|---|
| 5 |  |
| 10 |  |
| 20 |  |

**4. Border**

10) Border Enabled

Set whether the border should be displayed.

| State | Effect |
|---|---|
| Checked |  |
| Unchecked |  |

11) Border Color

Set Border Color

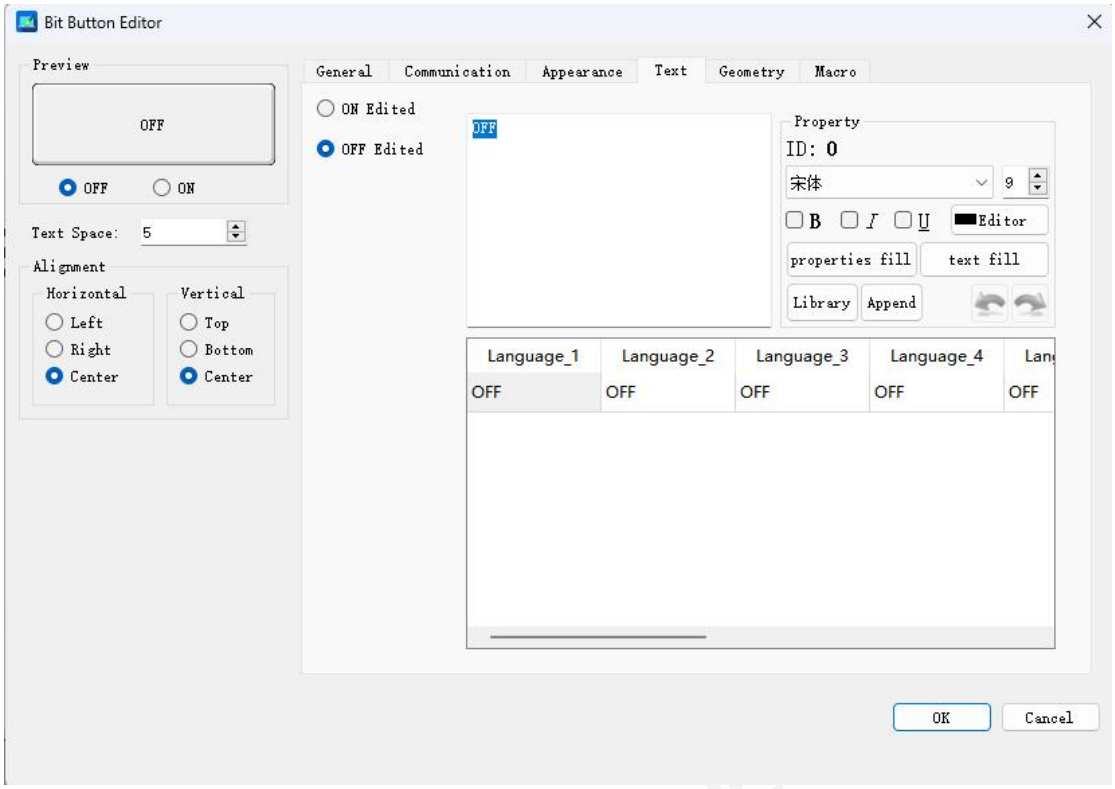
| Color | Effect |
| --- | --- |
| RGB(255,0,0) |  |
| RGB(0,0,0) |  |

12) Corner Radius

Set the button's corner radius.

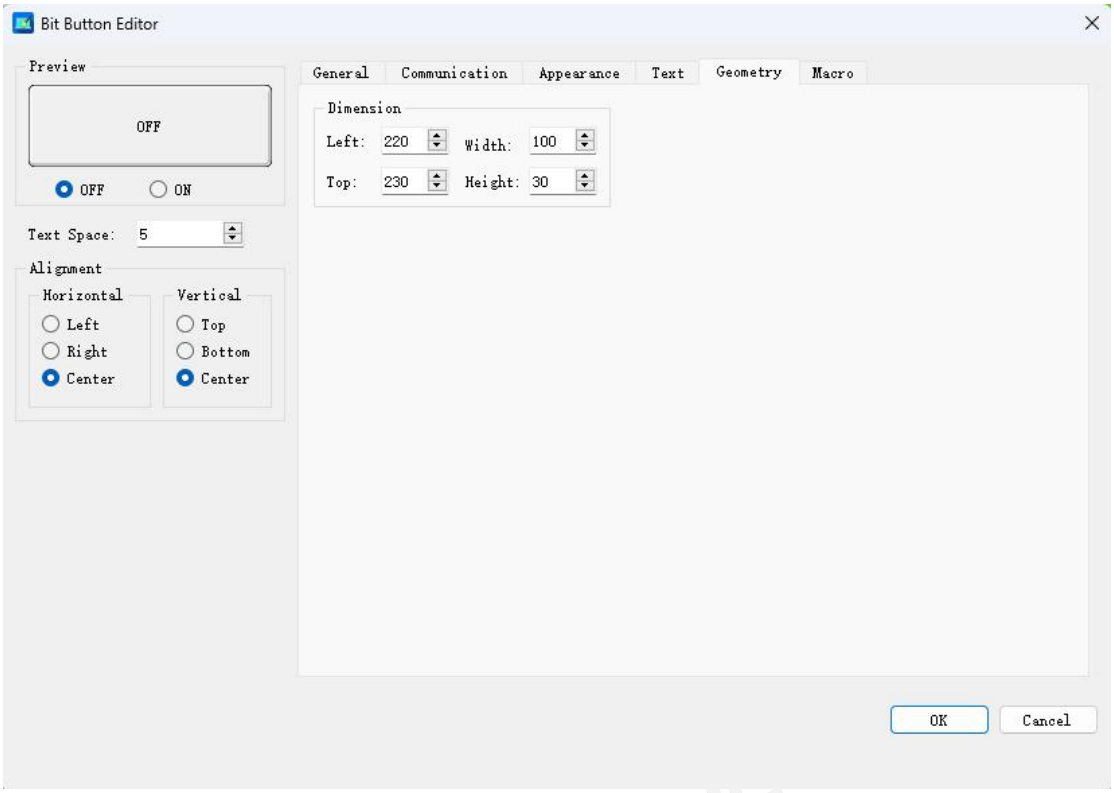
| Radius | Effect |
| --- | --- |
| 5 |  |
| 10 |  |
| 20 |  |

[Text]



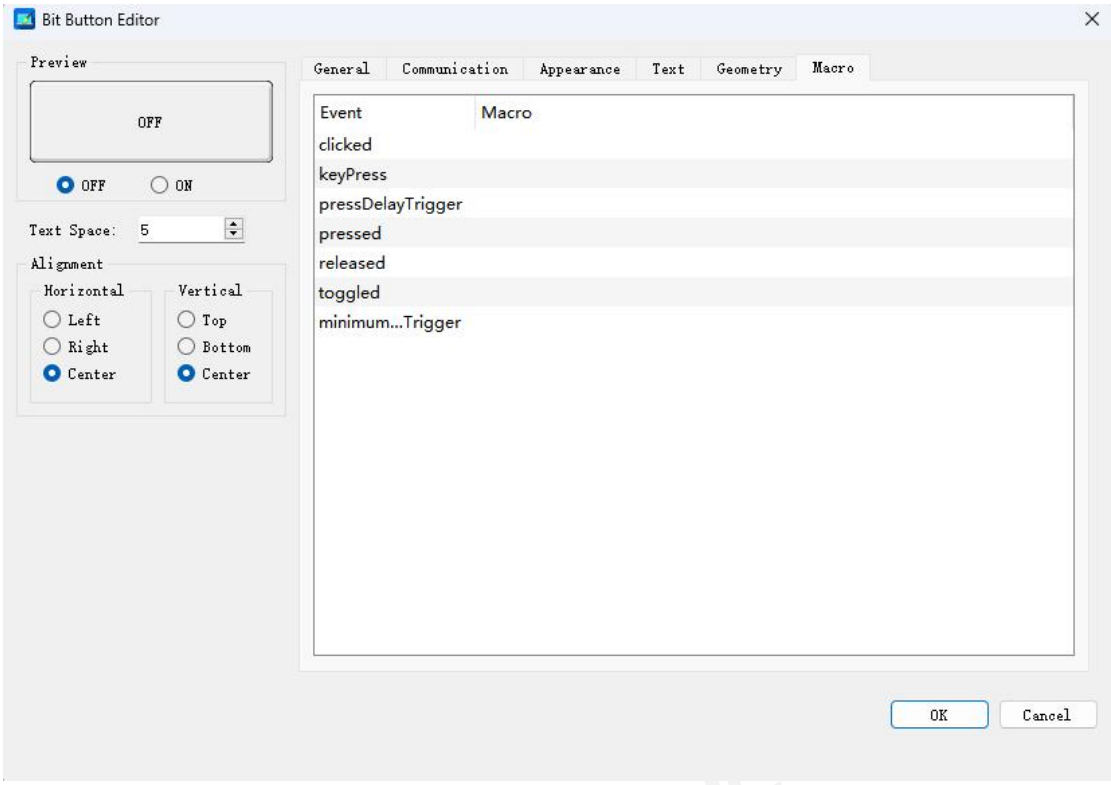

Set the text for the "Pressed" and "Released" states of the button. Text settings support multiple languages; refer to "Multilingual Editing" for details.

[Geometry Size]




This property controls the position and dimensions of the component within its parent window. Detailed parameter explanations are available under "Geometry Size".

[Macro]




| Events | Trigger Processes |
|---|---|
| MouseClick | Press the button using the mouse or keyboard shortcut, then release the button. |
| Key Pressed | Press the button using a keyboard shortcut. |
| Long Press Trigger | Press and hold the button using the mouse or keyboard shortcut for a period, combined with the delayed trigger function. |
| MousePress | Press the button using the mouse or keyboard shortcut. |
| MouseRelease | Press the button using the mouse or keyboard shortcut, then release the button. |
| Toggled | Toggle the button state when the button is in a checkable state. |

When triggering multiple events, the order of triggering events is as follows:

| Checked | Automatic | Click | Events |
|---|---|---|---|

| | repetition | | |
|---|---|---|---|
| Unchecked | Does not automatically repeat | Single click | MousePress<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>MouseRelease<br>MouseClick |
| | Automatic repetition | Single click | MousePress<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>MouseRelease<br>MouseClick<br>...（Loop） |
| Checkable | Does not automatically repeat | Single click | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | Automatic repetition | Single click | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>Toggled |

| | | | |
|---|---|---|---|
| | | | MouseRelease<br>MouseClick<br>...（Loop） |

## 6.14.4. Property Editor

The property editor provides a convenient way to set properties without needing to double-click to open the property settings dialog. Most properties in the editor function similarly to those in the settings dialog. Here, only functionalities not covered in the settings dialog are introduced.

### 1. Checkable


This property determines whether a component can be toggled. When enabled, users can switch its selected state by clicking on the component.

### 2. Checked


This property sets the button's toggle state. It takes effect when "Toggleable" is enabled.

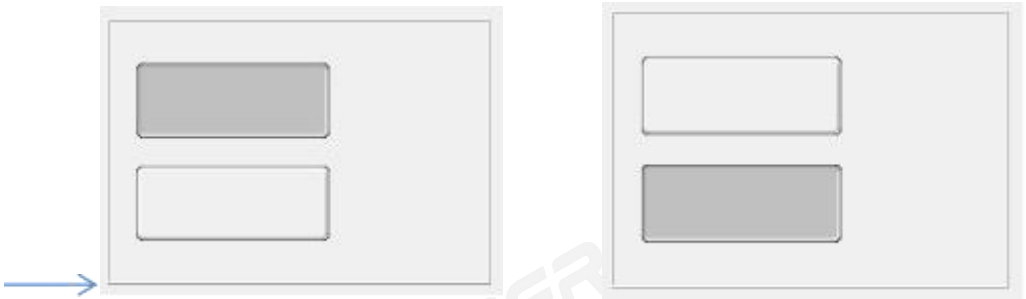
### 3. Auto Exclusive


If automatic exclusivity is enabled, checkable buttons within the same parent item can only have one button checked at a time. Selecting another button automatically deselects the previously checked button. This feature takes effect once "Checkable" is enabled.

Automatic Exclusive Selection Example:

Two buttons belong to the same parent component, and the "Checkable" property of the buttons is enabled. Pressing one button and then pressing another button will

automatically deselect the previously pressed button.



### 4. Auto Repeat

| | |
|---|---|
| autoRepeat | ☐ |

This property determines whether the button sends "pressed", "released", and "clicked" signals repeatedly when held down. The delay and repetition interval are defined by autoRepeatDelay and autoRepeatInterval in milliseconds.

### 5. Auto Repeat Delay

| | |
|---|---|
| autoRepeatDelay | 300 |

This property sets the delay time in milliseconds before auto-repeating actions (like "pressed", "released", and "clicked") begin after the button is initially pressed.

### 6. Auto Repeat Interval

| | |
|---|---|
| autoRepeatInterval | 100 |

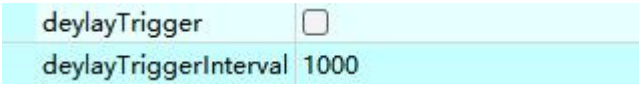
This property sets the interval time in milliseconds between auto-repeated actions while the button is held down.

### 7. Delayed Trigger

| | |
|---|---|
| deylayTrigger | ☐ |
| deylayTriggerInterval | 1000 |

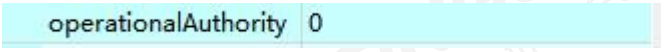
This property determines whether a "long-press trigger" action is activated when the button is released.

### 8. Delayed Trigger Time




This property sets the time in milliseconds after button release when the "long-press trigger" action is activated.

### 9. Operation Permission




Set the operational permissions for the component. If permissions are insufficient, the button component will be unable to operate.

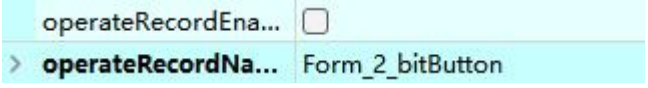
### 10. Enable


This property determines whether the button is enabled. When the component is disabled, it automatically ignores user input events such as click events, keyboard events, etc. Additionally, disabled controls are typically visually distinguished from enabled ones.

### 11. Channel Number Variable Control


When enabled, all associated variables have their channel number set to the actual value of Com40017.


### 12. Action Log

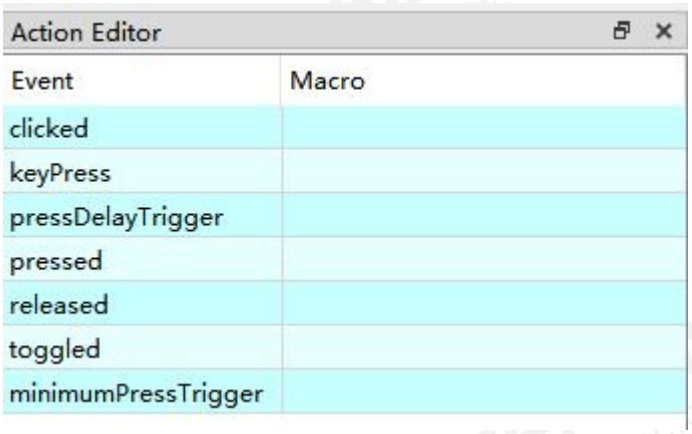



Enables logging of "Pressed" and "Released" actions of the toggle button to the action log.

**13. Action Log Name**


Sets the display name of the component in the action log table.

## 6.14.5. Action Editor




See details in "6.14.3.7 Macro" for more information.

## 6.14.6. Script Macros

For details, please refer to the "BitButton_Designer_CN.xml" file in the macroWizard folder.

## 6.15. Button Group

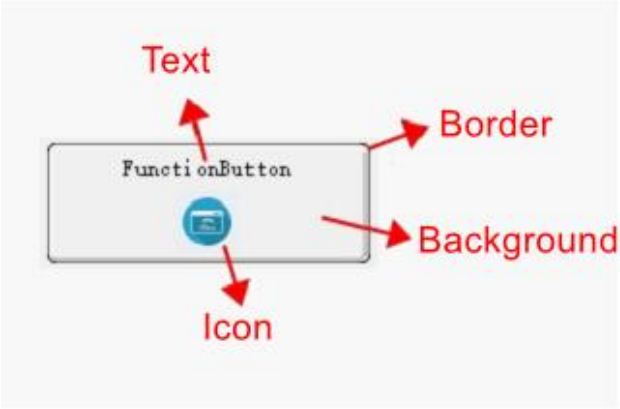
## 6.16. Check Box

## 6.17. Function Button

### 6.17.1. Function Description

1. Read and write variables.
2. Execute script macros by triggering events such as "click," "mouse down," and "mouse up."

### 6.17.2. Component Structure

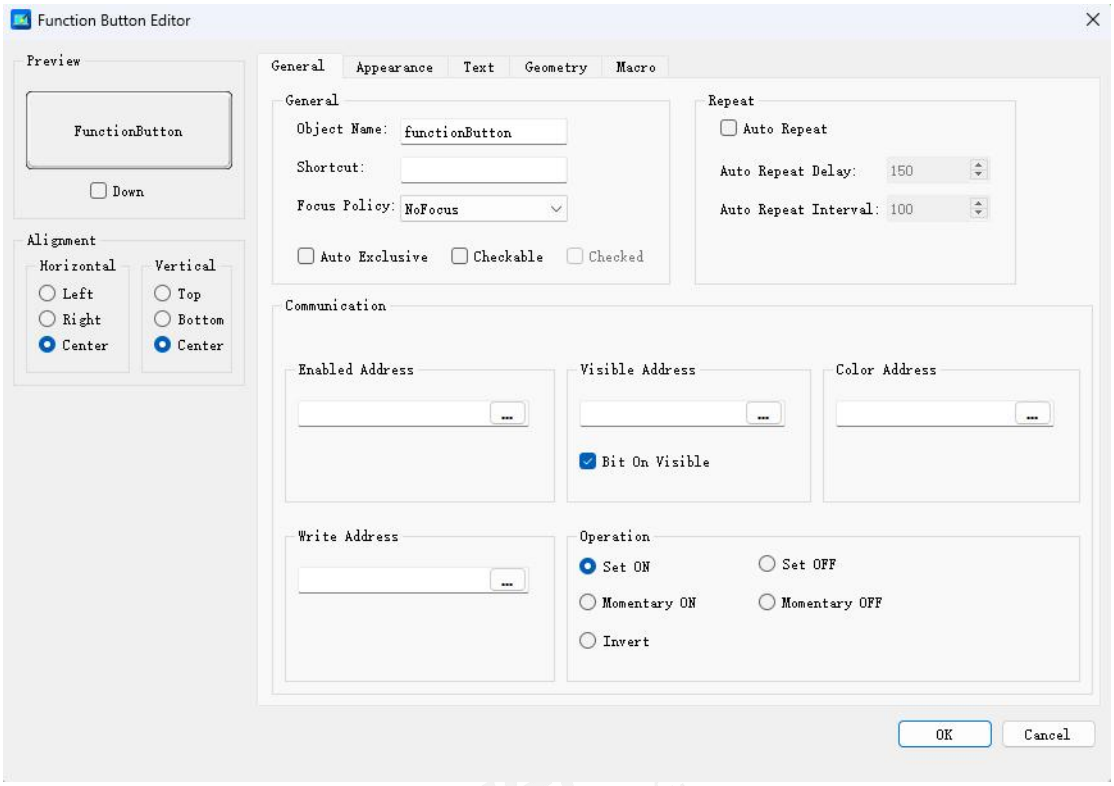



The function button component consists of "border," "text," "icon," and "background."

### 6.17.3. Property Settings Popup

In the HMI software editing interface, double-click the component with the left mouse button to bring up the property settings interface. The property settings popup includes five tabs: "General Properties," "Appearance," "Text," "Dimensions," and "Macros." Additionally, it contains "Preview" and "Align" settings.

## 6.17.3.1. Preview




### 1. Preview

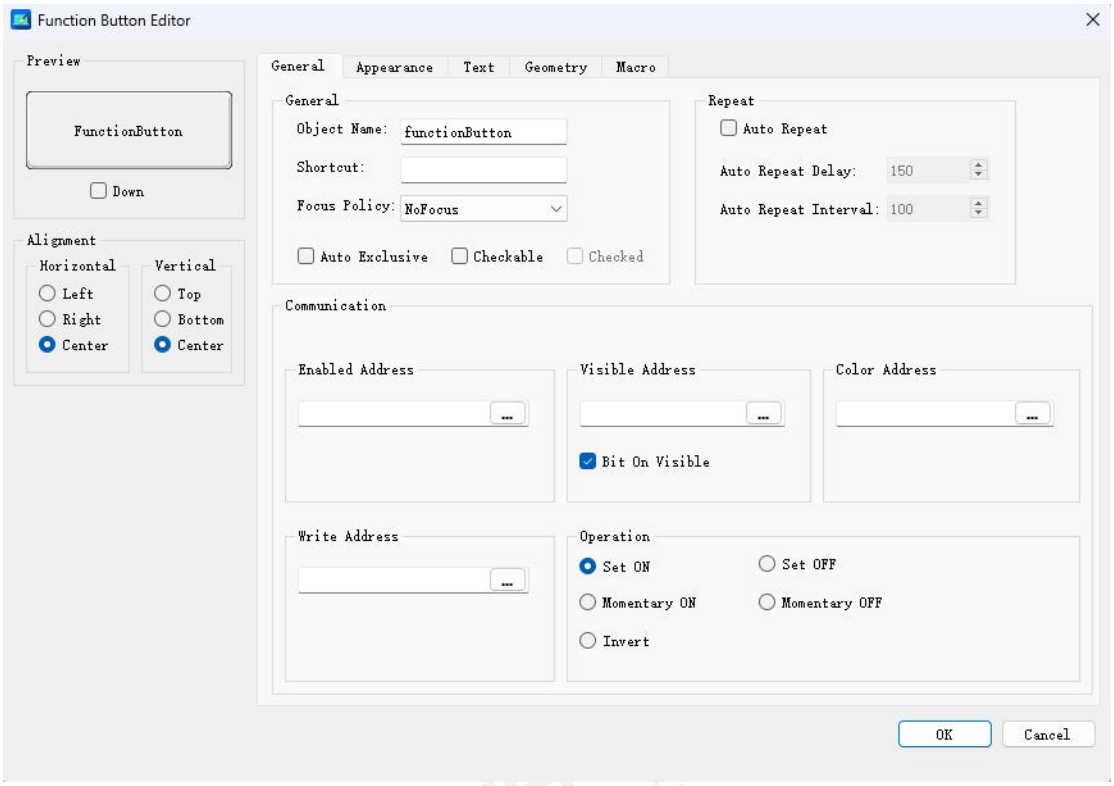
State OFF: Appearance of the preview button when in the OFF state.

State ON: Appearance of the preview button when in the ON state.

### 2. Align

Set the text alignment strategy. For details, see "Align." Note that once an icon is set, the text alignment strategy will be overridden, and the icon positioning strategy will take effect.

## 6.17.3.2. General Properties



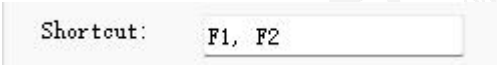

### 1. General Properties

1) Object Name

Each object can be identified and accessed by setting its Object Name. The Object Name is a string used to reference and manipulate the object in the code.

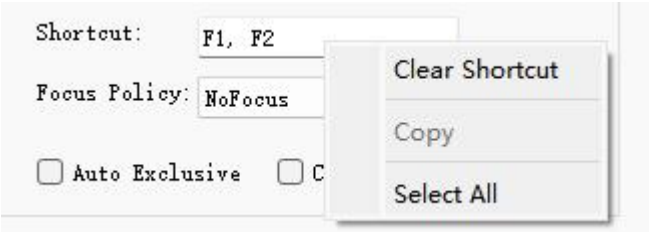
2) Shortcut Key

Shortcut keys provide a way to quickly access and execute specific functions or operations within the application without cumbersome mouse operations. Using shortcut keys can greatly improve user efficiency and ease of operation.




When multiple shortcut keys are set, any of them can trigger the button.

Right-clicking in the shortcut key input box will present an option to clear the shortcut key.




3) Focus Policy

The focus policy determines how a widget gains focus and handles focus events.

4) Auto-Exclusive

When auto-exclusivity is enabled, only one checkable button within the same parent can be checked at a time; checking another button will automatically uncheck the previous one. This takes effect after enabling "Checkable."

Example of auto-exclusivity:

Two buttons belong to the same parent widget, and their "Checkable" properties are enabled. Pressing one button and then pressing the other will automatically uncheck the first button.

5) Checkable

Sets whether the component can be checked. When a component is set to be checkable, the user can toggle its checked state by clicking on it.

6) Checked

This property sets the checked state of the button. It takes effect after enabling

"Checkable."

**2. Repeat**

7) Auto-Repeat

This property determines whether the button sends the pressed, released, and clicked signals repeatedly when it is pressed and held. The delay and repetition interval are defined by "Auto-Repeat Delay" and "Auto-Repeat Interval" in milliseconds.

8) Auto-Repeat Delay

This property sets the delay time for auto-repeat in milliseconds. After pressing the button, it enters the auto-repeat state after the delay time.

9) Auto-Repeat Interval

This property sets the interval for auto-repeat in milliseconds.

**3. System Variable Connection**

Use the "Variable Input Popup" to set the variables.

10) Control Enablement

This property sets whether the button is enabled. When the component is in a disabled state, it automatically ignores user input events such as click events and keyboard events. Additionally, disabled controls typically display a visually distinct disabled style to differentiate them from enabled controls.

| Bit Variable | Enabled State |
| --- | --- |
| On (i.e., bit variable is 1) | Enabled  |

| | |
|---|---|
| Off (i.e., bit variable is 0) |  <br>Note: This is not "pressed," but rather the effect of "not enabled." |

11) Control Visibility

This property sets whether the button is visible. When a control is set to visible, it will appear on the user interface. Conversely, when a control is set to invisible, it will be hidden from view, and users will not see it.

"Visible when bit is on": Sets the visibility of the button based on the associated bit variable's state.

| Visible when bit is on | Bit Variable State | Visibility State |
|---|---|---|
| 勾选 Checked | On (i.e., bit variable is 1) | Visible |
| | Off (i.e., bit variable is 0) | Invisible |
| 不勾选 Unchecked | On (i.e., bit variable is 1) | Invisible |
| | Off (i.e., bit variable is 0) | Visible |

12) Control Button Color

This property sets the button text and background colors based on the associated bit variable. When the bit variable value is 1, the "Color when bit is on" is used to set the background and foreground colors.

13) Write Address

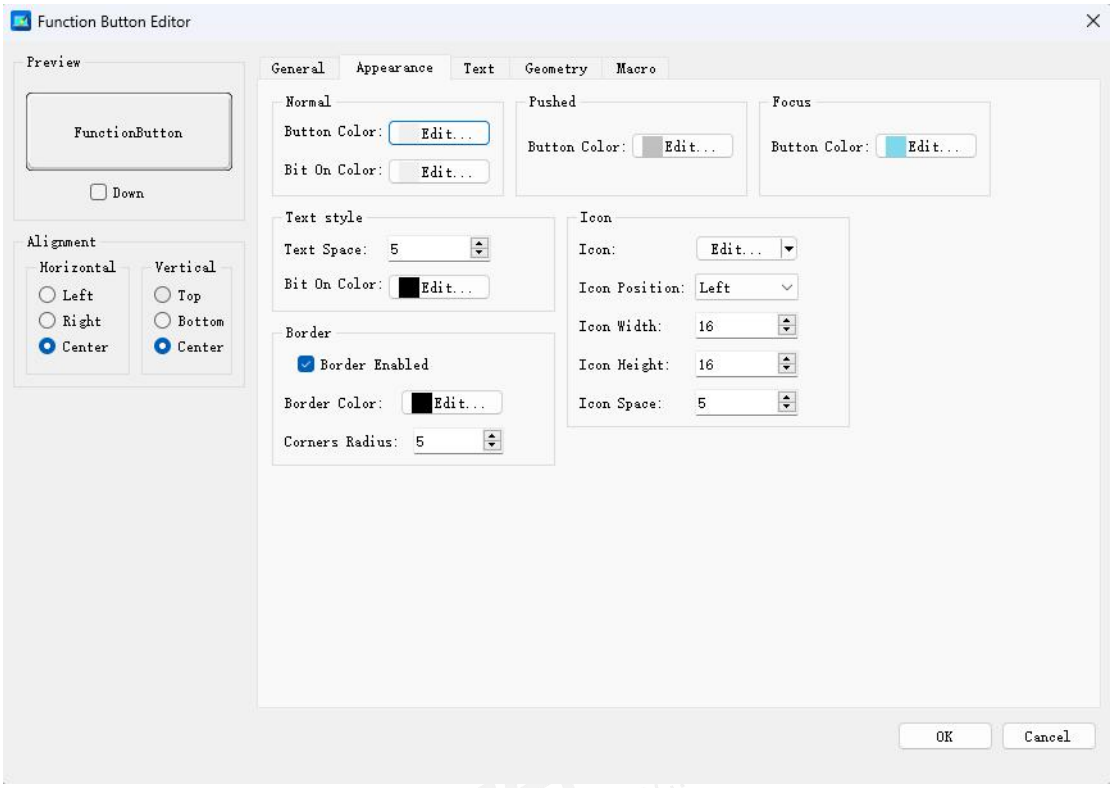
This property can set the button state associated with the bit variable. Depending on the "Pressed" and "Released" states of the button, the bit variable value is set. The value of the bit variable is related to the "Action" settings, for details see "Action".

14) Action

This property sets how the associated write address bit variable (write bit) changes when the button is pressed.

| Action | Bit Variable |
| --- | --- |
| Set to 1 | Press the button once, and the write bit remains in the on state continuously. |
| Set to 0 | Press the button once, and the write bit remains in the off state continuously. |
| Pressed 1 | When the button is pressed, the write bit remains in the on state continuously. When the button is released, the write bit remains in the off state continuously. |
| Pressed 0 | When the button is pressed, the write bit remains in the off state continuously. When the button is released, the write bit remains in the on state continuously. |
| Toggle | When the button is pressed, if the write bit is in the on state, change the write bit state to off; if the write bit is in the off state, change the write bit state to on. |

## 6.17.3.3. Appearance Shape




Button background color settings for various states have the following priority:

| Priority | Color Property |
|---|---|
| 1 | Pressed Background Color |
| 2 | Focused Background Color |
| 3 | Bit On Background Color |
| 4 | Released Background Color |

Button text color settings for two states have the following priority:

| Priority | Color Property |
|---|---|
| 1 | Text Color when Bit is On |
| 2 | Default Text Color |

**1. Normal**

1) Button Color

This property sets the background color of the button in the "Released" state.

2) Color when Bit is On

This property is related to the "Control Button Color" bit variable. When the bit variable value is 1, the button's background color changes to "Color when Bit is On"; otherwise, the button color is "Button Color".

**2. Pressed**

3) Button Color

This property sets the background color of the button in the "Pressed" state.

**3. Focused**

This property sets the background color of the button in the "Focused" state.

**4. Text Style**

4) Text Spacing

This property sets the spacing between the icon and the text.

5) Color when Bit is On

This property is related to the "Control Button Color" bit variable. When the bit variable

value is 1, the button text color changes to "Color when Bit is On".

**5. Border**

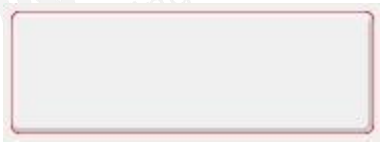
6) Border Visible

Sets whether the border is visible.

| State | Effect |
|---|---|
| Checked |  |
| Unchecked |  |

7) Border Color

Set the border color.

| Color | Effect |
|---|---|
| RGB(255,0,0) |  |
| RGB(0,0,0) |  |

8) Corner Radius

Set the corner radius of the button.

| Radius | Effect |
|---|---|
| 5 |  |

| | |
|---|---|
| 10 |  |
| 20 |  |

**6. Icon**

9) Icon

Set the icon for the function button. See the "Image Resources" instructions for how to add and clear images.

10) Icon Position

| Icon Position | Effect |
|---|---|
| Up |  |
| Down |  |
| Left |  |
| Right |  |

11) Icon Width

Set the width of the icon.

| Icon Height | Effect |
| --- | --- |
| 16 |  |
| 32 |  |
| 48 |  |

12) Icon Height

Set the height of the icon.

| Icon Height | Effect |
| --- | --- |
| 16 |  |
| 32 |  |
| 48 |  |

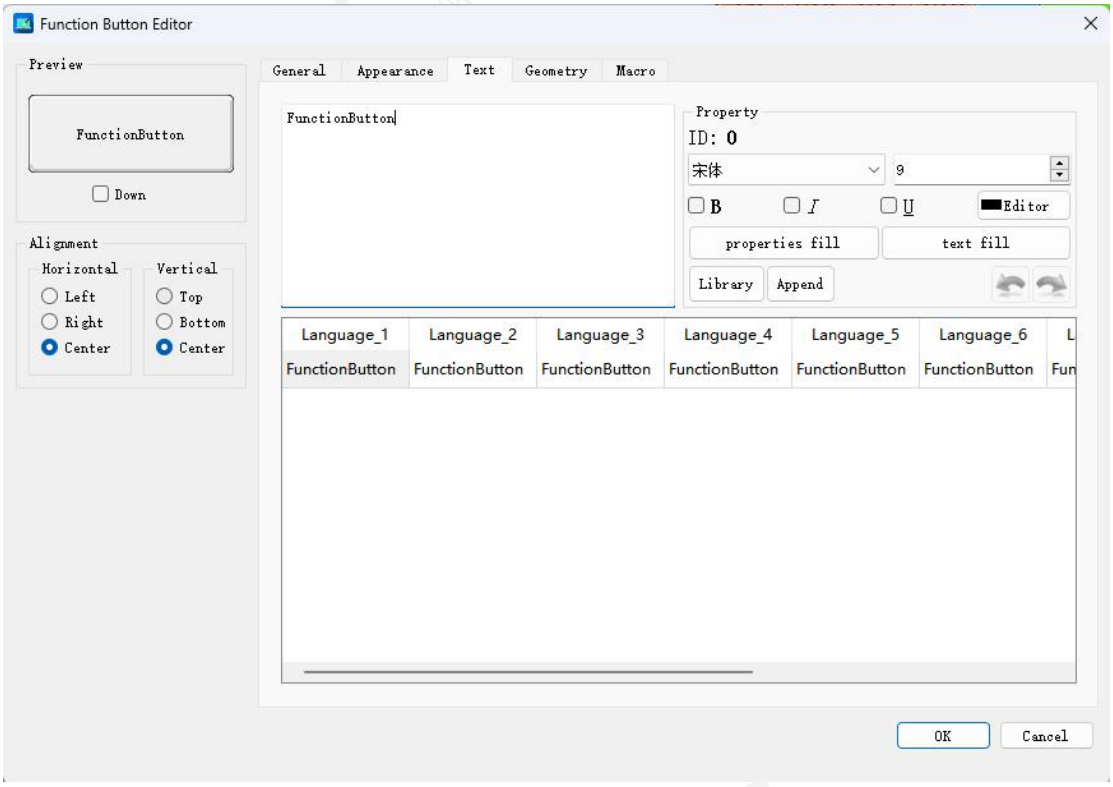
13) Icon Spacing

Set the spacing between the icon and the button border.

| Icon Spacing | Effect |
| --- | --- |
| 5 |  |

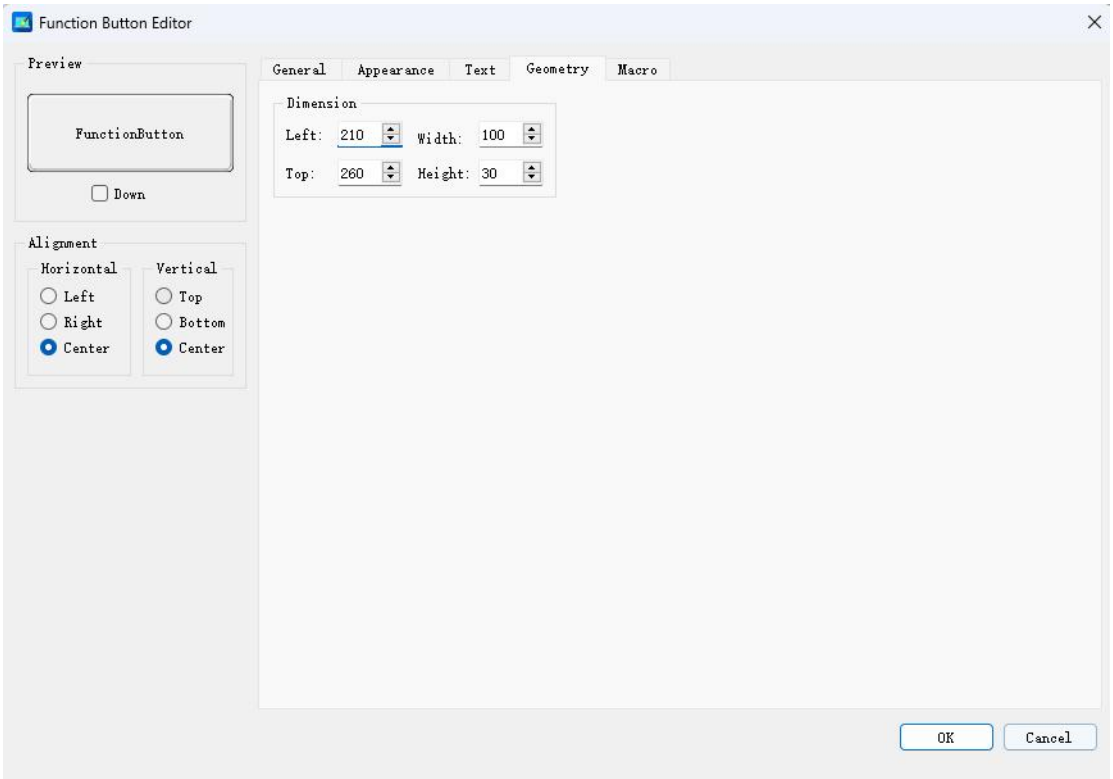
| | |
|---|---|
| 10 |  |
| 20 |  |

## 6.17.3.4. Text



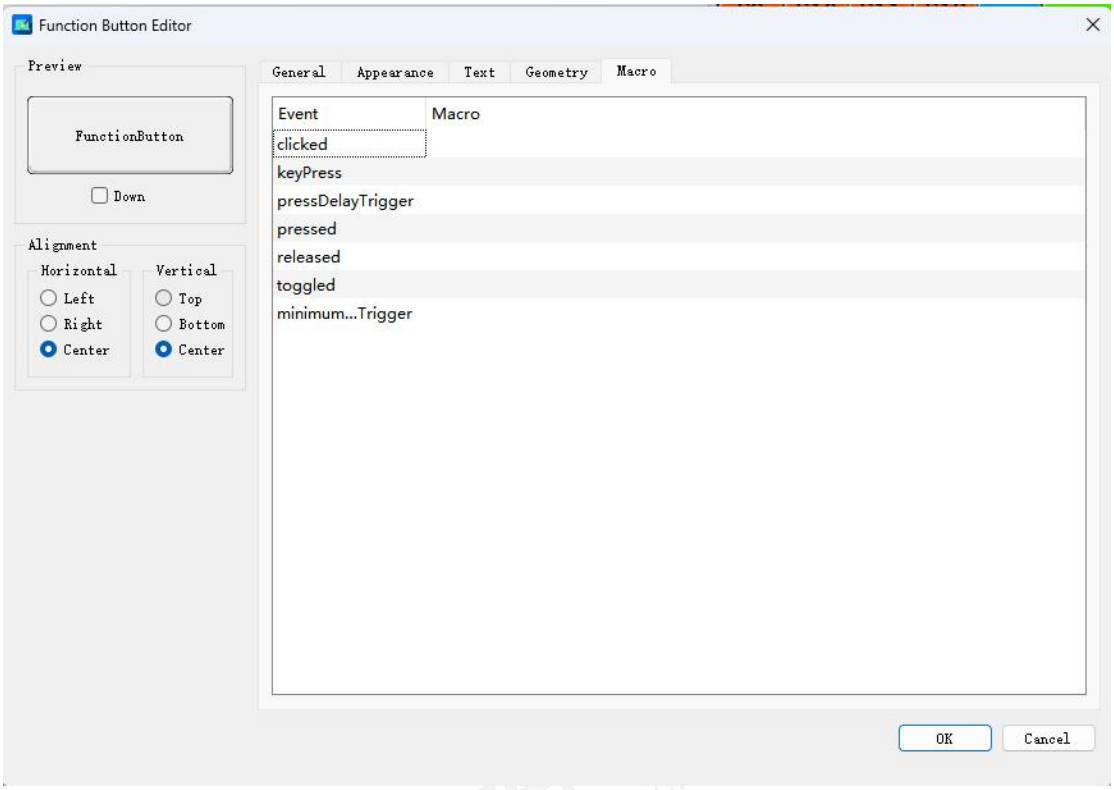

Set the text for the button. The text settings support multiple languages. For details, refer to "Multi-Language Editing".

## 6.17.3.5. Geometry Size




This property controls the position and size of the component within the parent window. For details on each parameter, refer to "Geometry Size".

## 6.17.3.6. Events




| Events | Trigger Process |
|---|---|
| MouseClick | Press the button with the mouse or shortcut keys, then release the button. |
| Key Press | Press the button with shortcut keys. |
| Long Press Trigger | Press the button with the mouse or shortcut keys and hold it for a period of time, used in conjunction with delayed trigger functionality. |
| MousePress | Press the button with the mouse or shortcut keys. |
| MouseRelease | Press the button with the mouse or shortcut keys, then release the button. |
| Toggled | If the button is in a checkable state, toggling the button state. |

When triggering multiple events, the sequence of events is as follows:

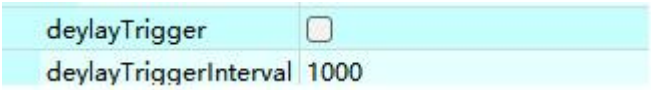
| Checked | Repeat | Action | Events |
|---|---|---|---|
| Unchecked | Not auto-repeat | Click | MousePress<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>MouseRelease<br>MouseClick |
| | Auto-repeat | Click | MousePress<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>MouseRelease<br>MouseClick<br>...（Loop） |
| Checked | Not auto-repeat | Click | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | Auto-repeat | Click | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress |

| | | | |
|---|---|---|---|
| | | | Toggled<br>MouseRelease<br>MouseClick<br>...（Loop） |

## 6.17.4. Property Editor

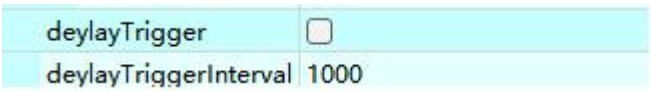
The property editor provides a convenient way to set properties without needing to double-click to open the property settings popup. Most of the properties in the property editor have the same functions as those in the property settings popup. Only features not covered in the property settings popup are introduced here.

### 1. Delayed Trigger



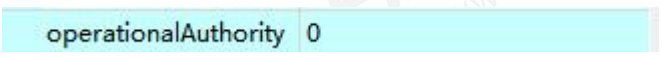

This property sets whether the "Long Press Trigger" action is triggered when the button is released.

### 2. Delayed Trigger Time




This property sets the delay time in milliseconds after button release before triggering the "Long Press Trigger" action.

### 3. Operation Permissions

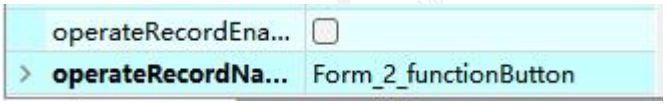



Set the operational permissions for the component. If permissions are insufficient, the button component will be unable to operate.

#### 4. Enabled


This property sets whether the button is enabled. When the component is disabled, it automatically ignores user input events such as clicks and keyboard events. Additionally, disabled controls typically display a visually distinct disabled style to differentiate them from enabled controls.

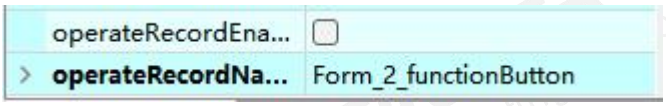
#### 5. Operation Logging




Enable operation logging for the component.

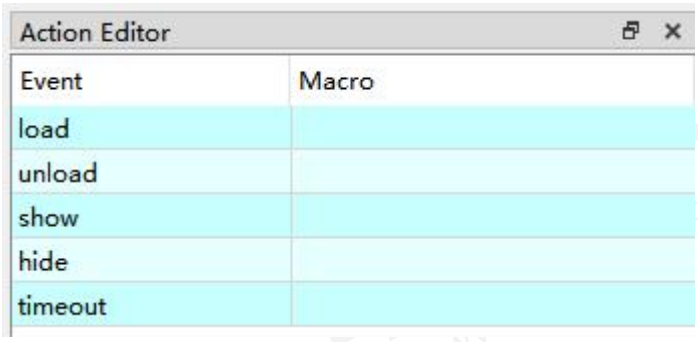
Record the "Pressed" and "Released" actions of bit buttons in the operation log.

#### 6. Operation Log Name




Set the display name of the component in the operation log table.

## 6.17.5. Action Editor




For details, see the explanation in "6.17.3.6 Macros".

### 6.17.6. Script Macros

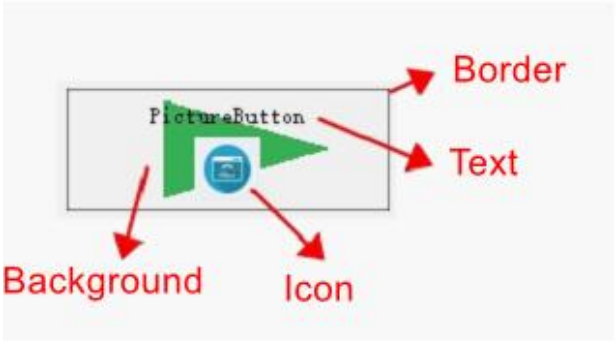
For details, refer to the "FunctionButton_Designer_CN.xml" file in the macroWizard folder.

## 6.18. Image Button

### 6.18.1. Feature Introduction

1. Executes script macros by triggering events such as "Click", "Mouse Press", and "Mouse Release".
2. Used as function keys for virtual keyboards.

### 6.18.2. Component Structure

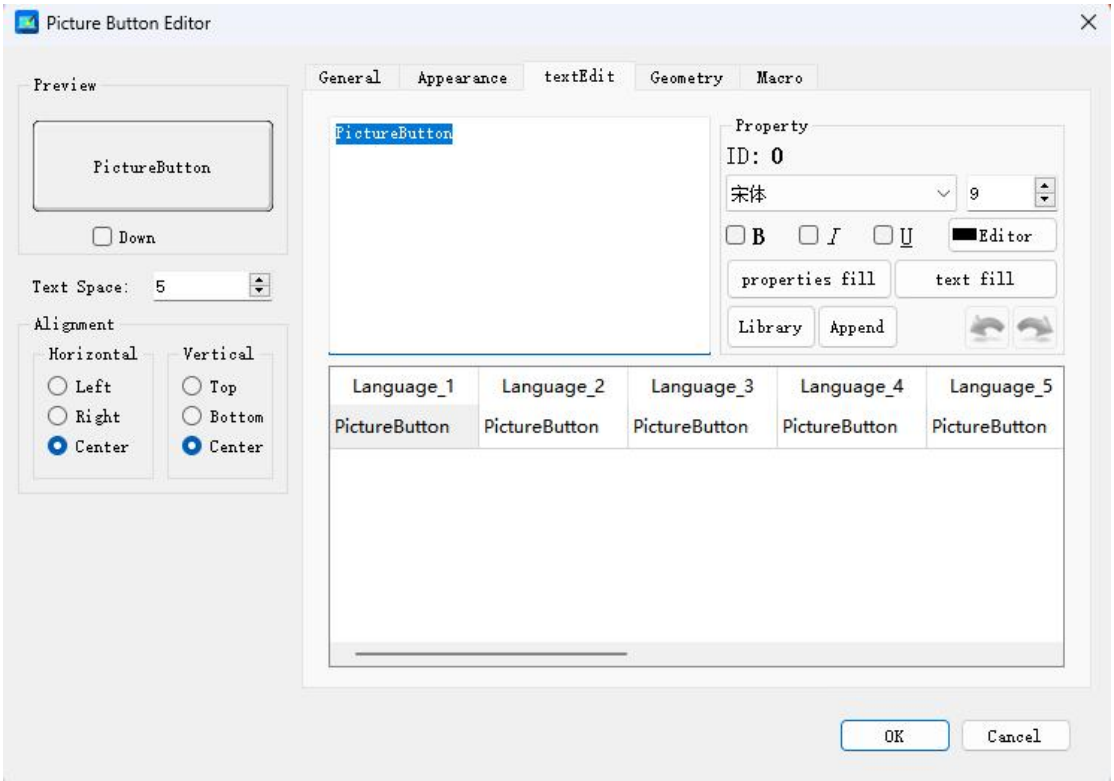



The image button component consists of "Border", "Text", "Icon", and "Background".

### 6.18.3. Property Settings Popup

In the HMI software editing interface, double-click the component with the left mouse button to open the property settings popup. The property settings popup includes "General Properties", "Appearance Shape", "Text Settings", "Geometry Size", "Macros", totaling five pages. It also includes settings for "Preview", "Text Spacing", and "Alignment".

## 6.18.3.1. Preview




**1. Preview**

**State: OFF**

Appearance of the button when it is in the OFF state.

**State: ON**

Appearance of the button when it is in the ON state.

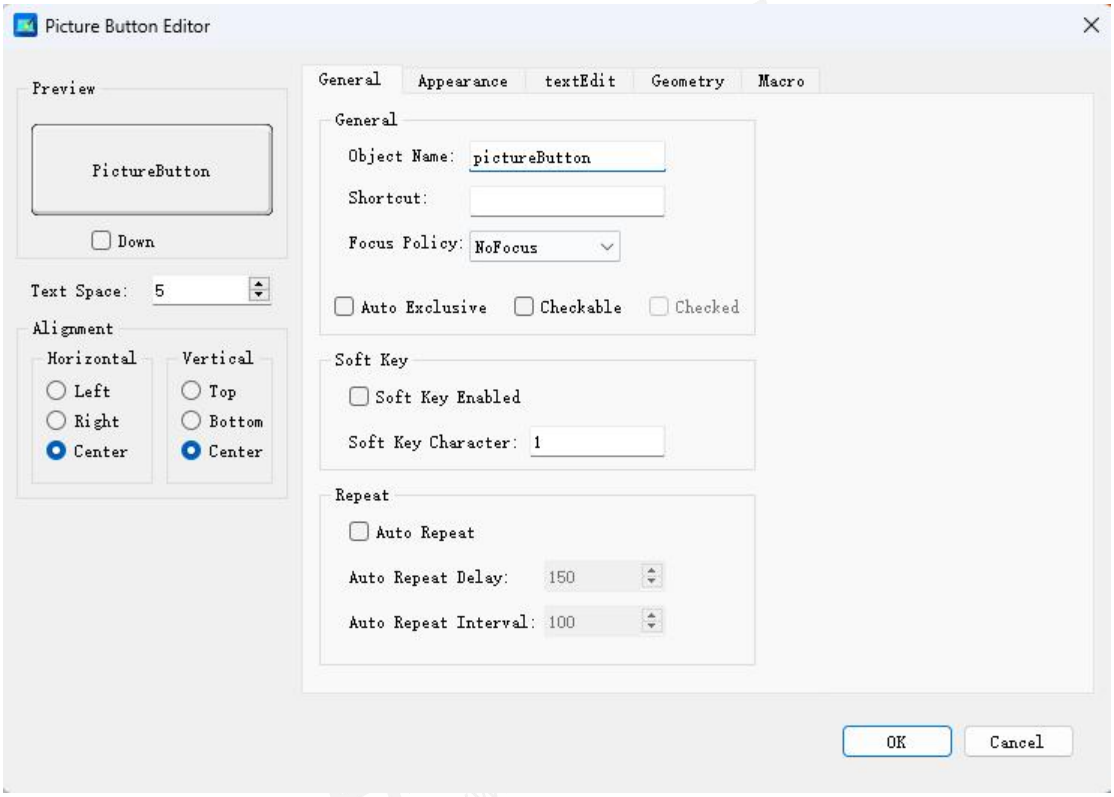
**2. Text Spacing**

Set the spacing between the button icon and text.

**3. Alignment**

Set the text alignment strategy. For details, refer to "Alignment". It's important to note that once an icon is set, the text alignment strategy becomes ineffective, and the icon positioning strategy takes effect.

## 6.18.3.2. General Properties



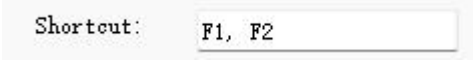

### 1. General Properties

1) Object Name

Each object can be identified and accessed by setting an Object Name. The Object Name is a string used to reference and manipulate objects in code.

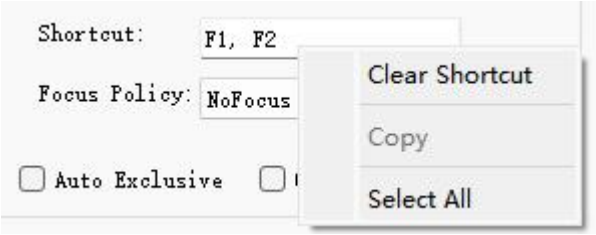
2) Shortcut Keys

Shortcut keys in applications provide a quick way to access and execute specific functions or operations without the need for cumbersome mouse operations. Using shortcut keys can significantly enhance user efficiency and operational convenience.




When setting multiple shortcut keys, all configured shortcuts can trigger the button.

Right-clicking in the shortcut key input box provides an option to clear the shortcut keys.




3) Focus Policy

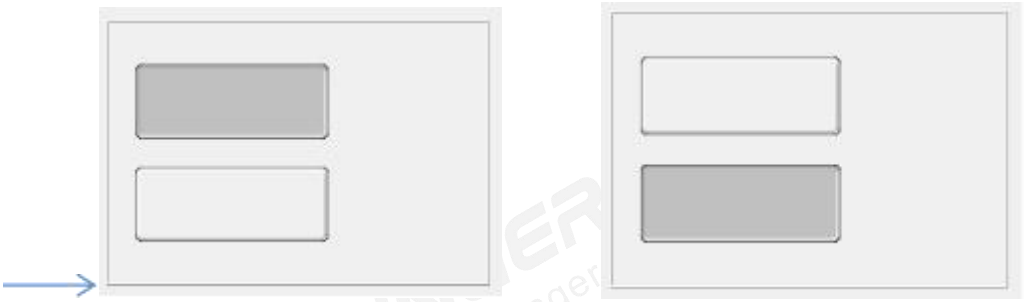
Focus Policy determines how a widget (component) acquires focus and handles focus events.

4) Automatic Exclusive

Enabling Automatic Exclusive means that among checkable buttons belonging to the same parent, only one button can be checked at a time. Checking another button automatically unchecks the previously checked one. This feature is activated when "Checkable" is enabled.

Example of Automatic Exclusive:

Two buttons belong to the same parent component, and the "Checkable" property of the buttons is enabled. Pressing one button and then pressing another button will automatically uncheck the previously pressed button.



5) Checkable

Set whether the component is checkable. When a component is set to be checkable, users can toggle its checked state by clicking the component.

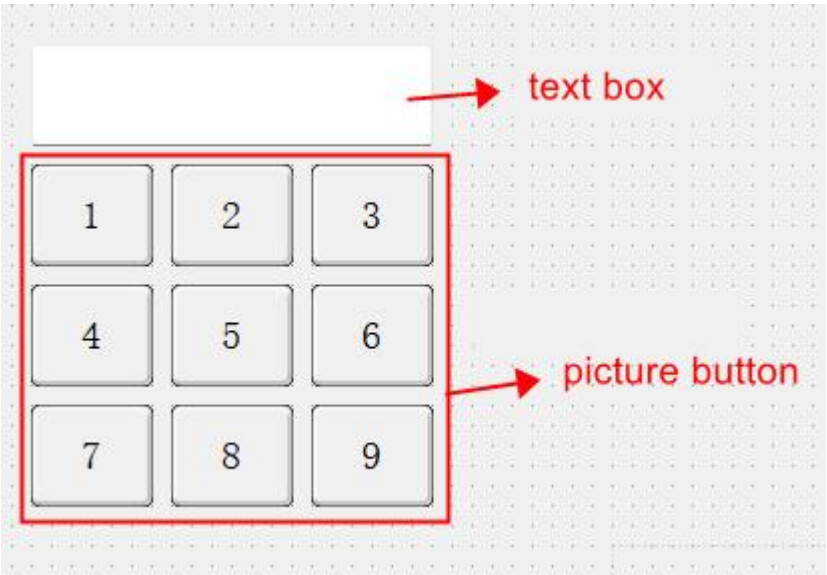
6) Checked

This property sets the checked state of the button. It becomes effective when "Checkable" is enabled.

## 2. Soft Key Function

Image buttons can be used as function keys on a virtual keyboard.

As shown in the example below, enable the "Use Soft Key Function" for each image button and set the "Soft Key Symbol" sequentially as "1", "2", "3", "4", "5", "6", "7", "8", and "9". When the buttons are clicked during runtime, the "Soft Key Symbol" will be input into the current focus component (text box).




7) Use Soft Key Function This property sets whether the image button enables the "Soft Key Function".

8) Soft Key Symbol This property sets the corresponding ASCII character for the image button when the "Soft Key Function" is enabled.

## 3. Repeat

9) Auto Repeat

This property determines whether the button repeatedly sends pressed, released, and clicked signals when it is pressed and held. The delay and repeat interval are defined by "Auto Repeat Delay" and "Auto Repeat Interval" in milliseconds.

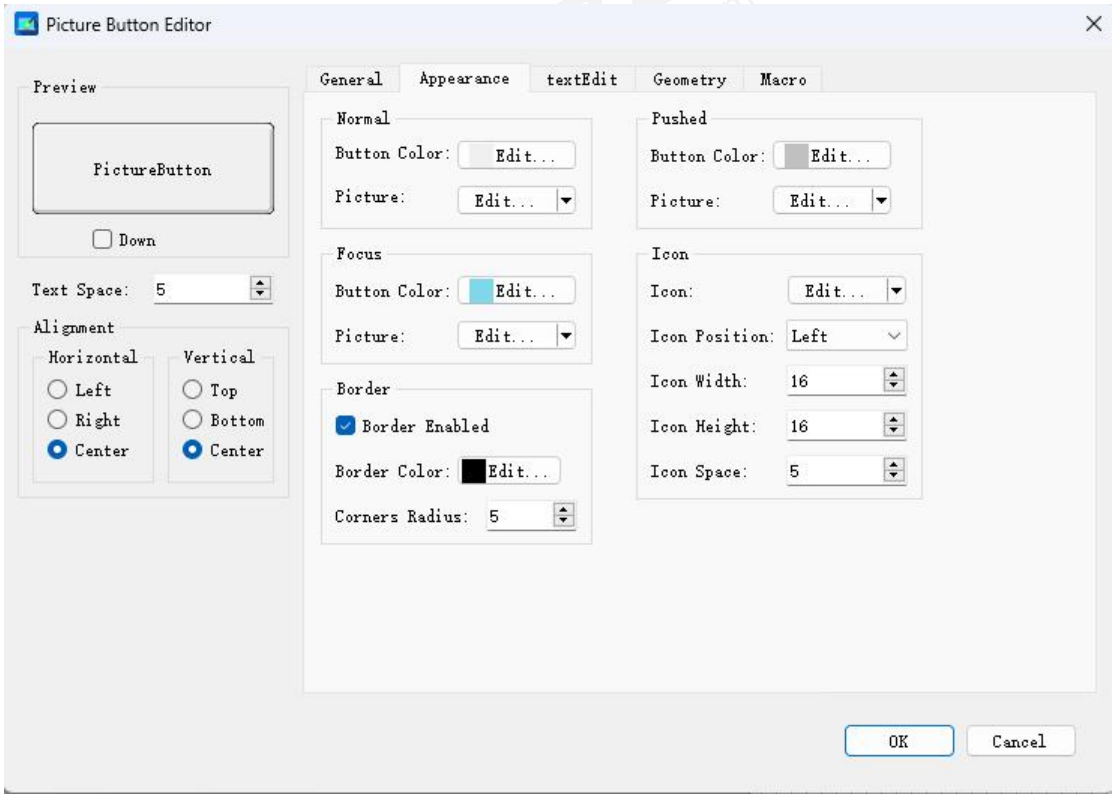
10) Auto Repeat Delay

This property sets the delay time in milliseconds for auto repeat. After the button is pressed and the delay time has elapsed, the button enters the auto repeat state.

11) Auto Repeat Interval

This property sets the interval for auto repeat in milliseconds.

## 6.18.3.3. Appearance Shape




The button has background color settings for various states, with the following priority for

background colors:

| Priority | Color Property |
|---|---|
| 1 | Background Color When Pressed |
| 2 | Background Color When Focused |
| 3 | Background Color When Released |

Background Image Settings for Button in Various States, with the Following Priority

| Priority | Color Property |
|---|---|
| 1 | Background Color When Pressed |
| 2 | Background Color When Focused |
| 3 | Background Color When Released |

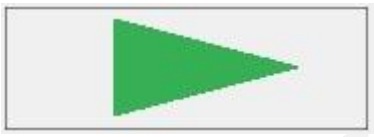
**1. Normal**

1) Button Color

This property sets the background color of the button when it is in the "released" state.

2) Image

Sets the background image of the button when it is in the "released" state. For information on how to add and clear images, refer to the "Image Resources" section.



“图片”属性会覆盖“按钮颜色”属性

**2. Pressed**

3) Button Color

This property sets the background color of the button when it is in the "pressed" state.

4) Image

This property sets the background image of the button when it is in the "pressed" state.

The "Image" property overrides the "Button Color" property.

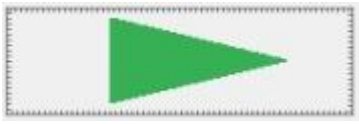
### 3. Focus

5) Button Color

This property sets the background color of the button when it is in the "focus" state.

6) Image

This property sets the background image of the button when it is in the "focus" state.



The "Image" property overrides the "Button Color" property.

### 4. Icon

7) Icon

Sets the icon for the image button. For information on how to add and remove images, refer to the "Image Resources" section.

8) Icon Position

| Icon Position | Effect |
|---|---|
| Up |  |
| Down |  |
| Left |  |
| Right |  |

9) Icon Width

Set the width of the icon.

| Icon width | Effect |
|---|---|
| 16 |  |
| 32 |  |
| 48 |  |

10) Icon height

Set the height of the icon.

| Icon height | Effect |
|---|---|
| 16 |  |
| 32 |  |
| 48 |  |

11) Icon spacing

Set the spacing between the icon and the button border

| Icon spacing | Effect |
|---|---|
| 5 |  |
| 10 |  |
| 20 |  |

**5. Border**

12) Border enabled

Set whether to display the border.

| State | Effect |
| --- | --- |
| Checked |  |
| Unchecked |  |

13) Border color

Set the border color.

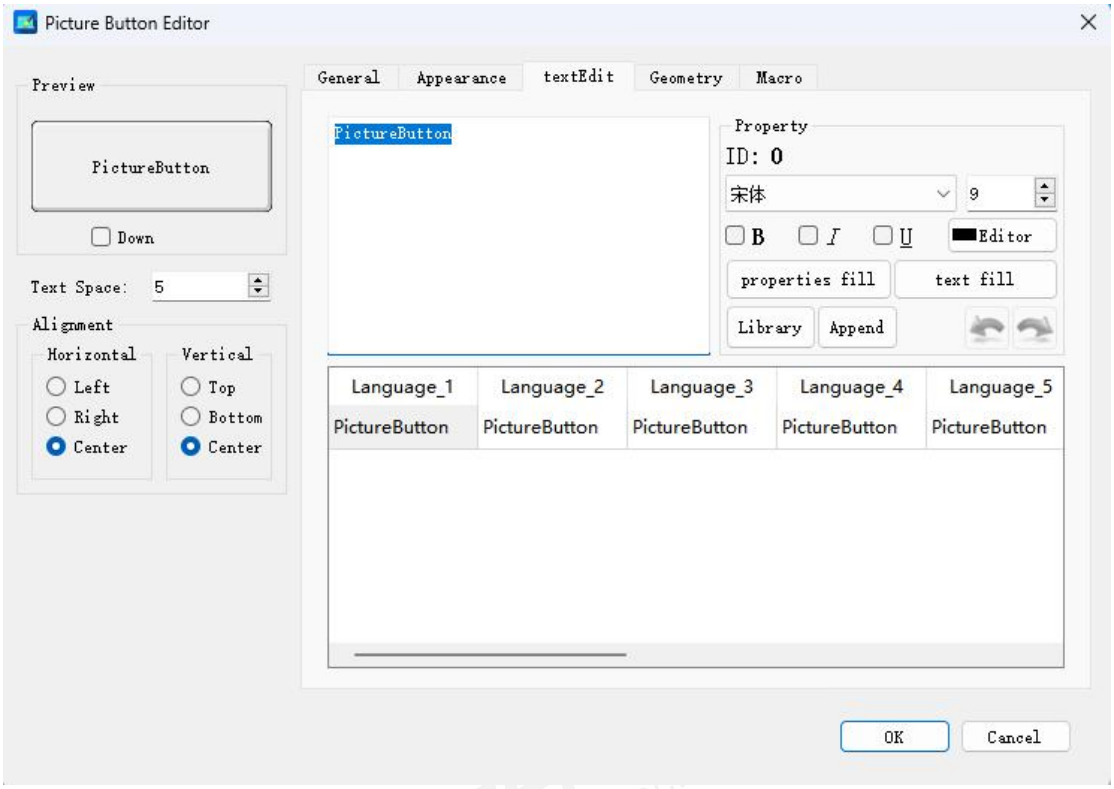
| Color | Effect |
| --- | --- |
| RGB(255,0,0) |  |
| RGB(0,0,0) |  |

14) Border radius

Set the border radius of the button.

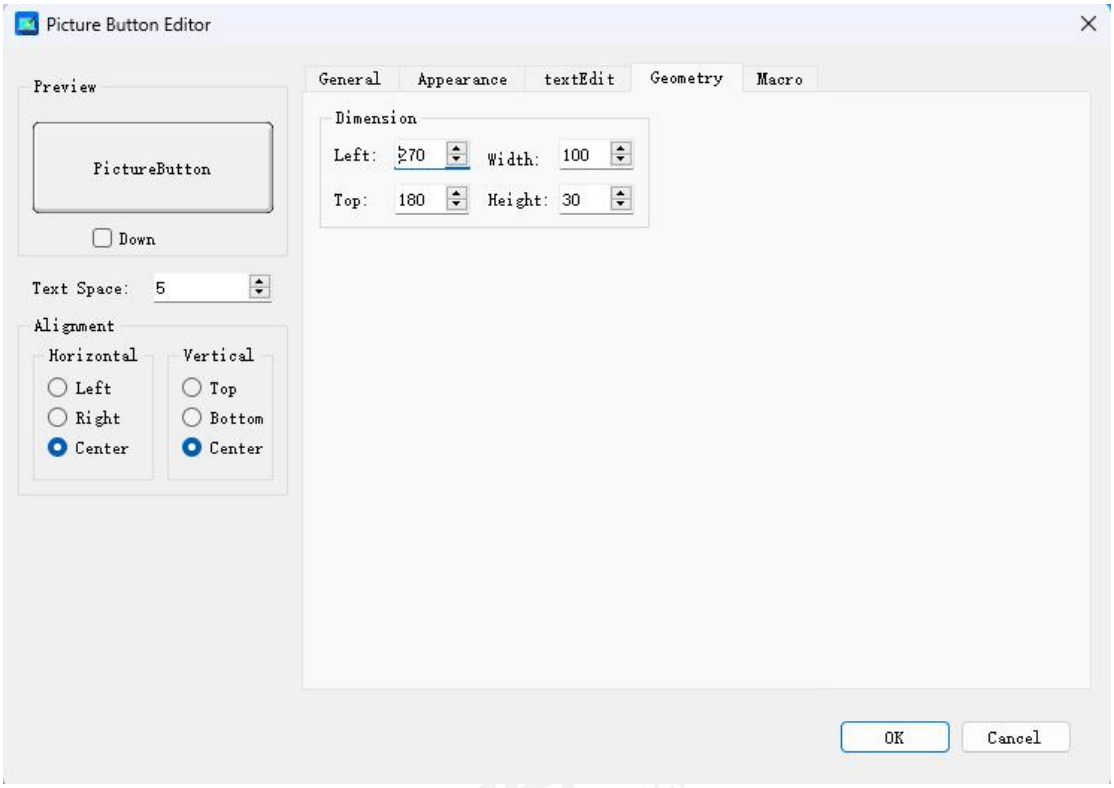
| Radius | Effect |
| --- | --- |
| 5 |  |
| 10 |  |
| 20 |  |

## 6.18.3.4. Text Settings



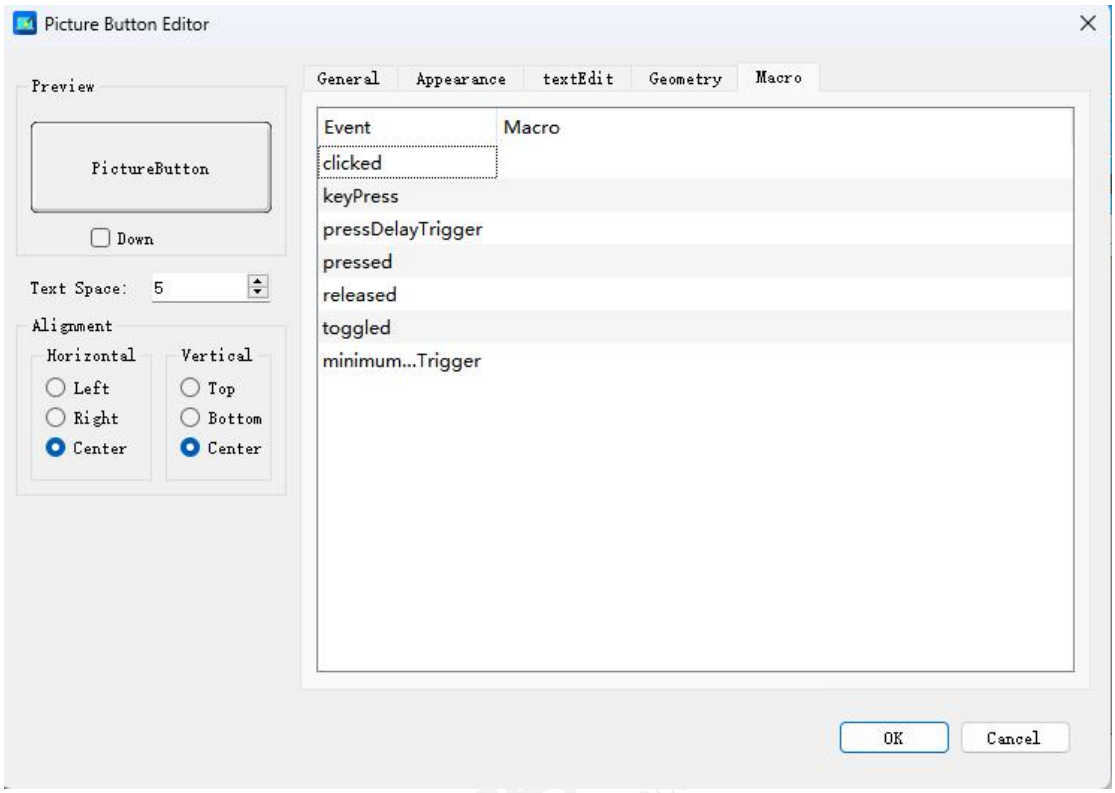

Set the text for the button. Text settings support multiple languages, details can be found in "Multilingual Editing".

## 6.18.3.5. Geometric Dimensions




This property controls the position and size of the component within its parent window. Detailed parameter explanations can be found in "Geometric Dimensions".

## 6.18.3.6. Macro



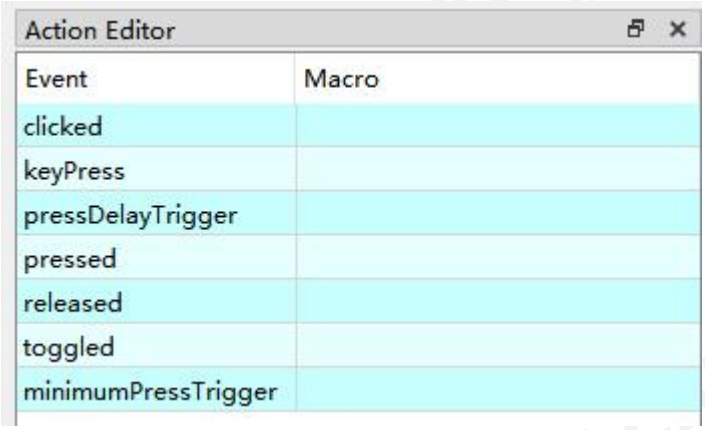

| Events | Trigger Process |
|---|---|
| MouseClick | Press the button with the mouse or shortcut keys, then release the button. |
| Key press | Press the button using shortcut keys. |
| Long press trigger | Press the button with the mouse or shortcut keys and hold it for a period of time, used in conjunction with delayed trigger functionality. |
| MousePress | Press the button with the mouse or shortcut keys. |
| MouseRelease | Press the button with the mouse or shortcut keys, then release the button. |
| Toggled | The button is in a toggleable state, toggle the button state. |

When triggering multiple events, the order of triggering events is as follows:

| Checked | Repeat | Operations | Trigger Process |
|---|---|---|---|
| Not checkable | Do not auto-repeat | Click | MousePress<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>MouseRelease<br>MouseClick |
| | Auto-repeat | Click | MousePress<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>MouseRelease<br>MouseClick<br>...（Loop） |
| Checkable | Do not auto-repeat | Click | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | Auto-repeat | Click | MousePress<br>Toggled<br>MouseRelease<br>MouseClick |
| | | Long press | MousePress<br>Toggled |

| | | | |
|---|---|---|---|
| | | | MouseRelease<br>MouseClick<br>...（Loop） |

## 6.18.4. Action Editor

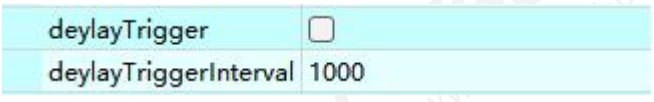



See details in "6.18.3.6 Macro".

## 6.18.5. Property Editor

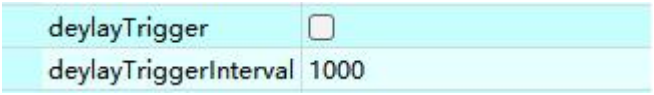
The Property Editor provides a convenient way to set properties without needing to double-click to open a property settings dialog. Most properties in the Property Editor function the same as in the property settings dialog. Here, only features not covered in the property settings dialog will be introduced.

### 1. Delayed Trigger



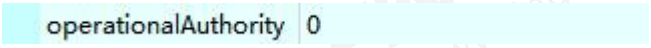

This property determines whether the "long press trigger" action is triggered when the button is released.

### 2. Delayed Trigger Time




This property sets the duration in milliseconds after which the "long press trigger" action is triggered when the button is released.

### 3. Operation Permission

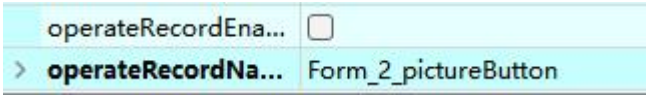



Set the operational permissions for the component. If permissions are insufficient, the button component will be unable to operate.

### 4. Enabled

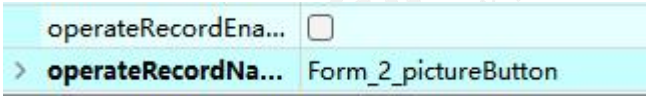

This property sets whether the button is enabled. When a component is disabled, it automatically ignores user input events such as clicks and keyboard events. Additionally, disabled controls are typically visually distinguished from enabled ones.

### 5. Operation Logging




Enable operation logging for the component. Record the "press" and "release" actions of the image button in the operation log.

### 6. Operation Log Name




Set the display name of the component in the operation log table.

## 6.18.6. Script Macros

For details, refer to "PictureButton_Designer_CN.xml" in the macroWizard folder.

## 6.19. Radio Buttons

## 6.20. CadWidget（CAD）

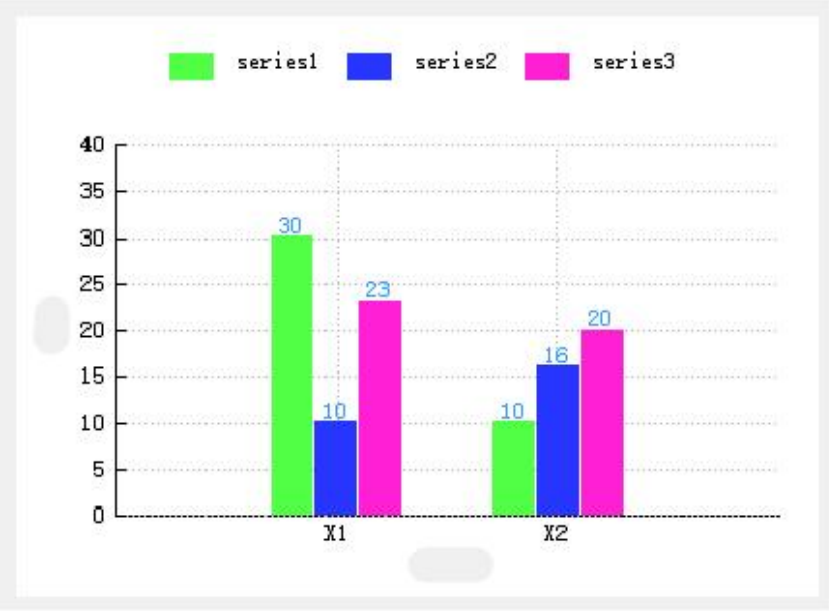
Refer to the following documentation for details:

二维 CAD/CAM 组件使用指南

## 6.21. Bar Chart

**Function Introduction**

The bar chart component is used in HMIs to visually represent statistical reports of data in a bar form. It consists of a series of vertical (or horizontal) bars of varying heights (or lengths) that depict data distribution. Bar charts are typically used for analyzing smaller datasets. They can also be displayed horizontally, which provides users with an intuitive view of the data.



## 6.21.1. General Properties

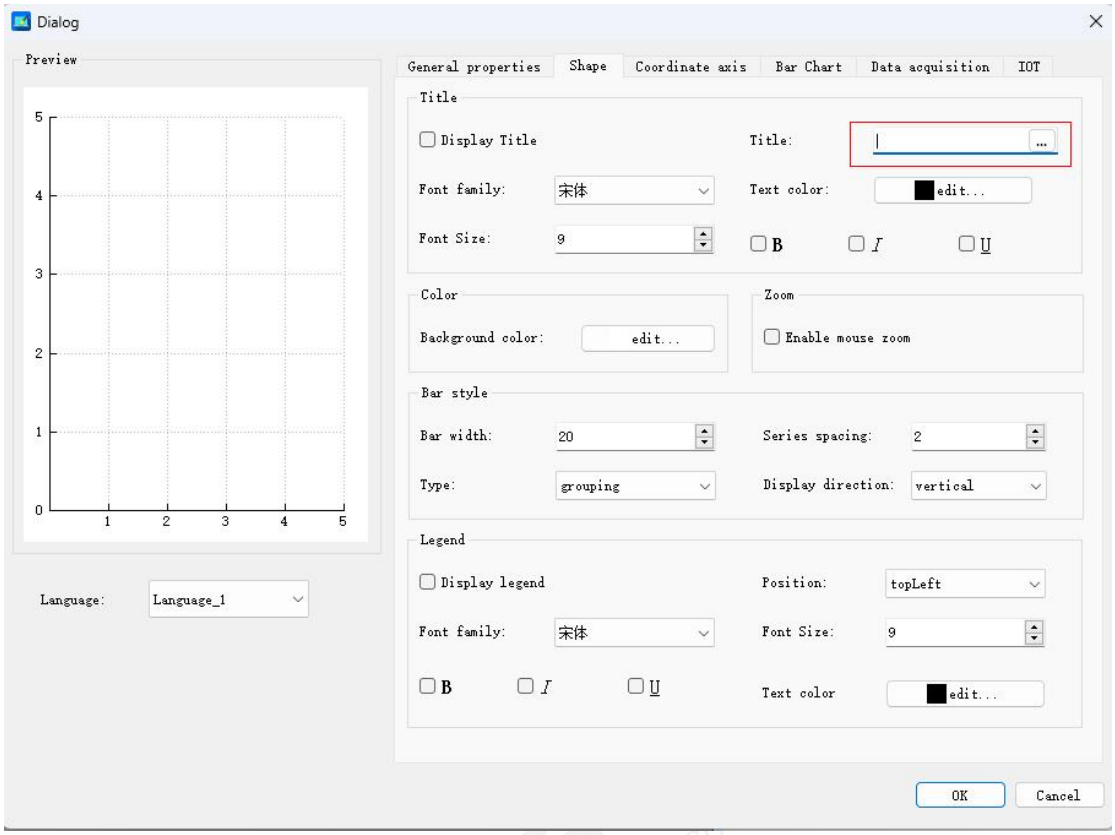
View common property descriptions for Object Name, Focus Policy, and Geometric Dimensions.

## 6.21.2. Appearance Settings

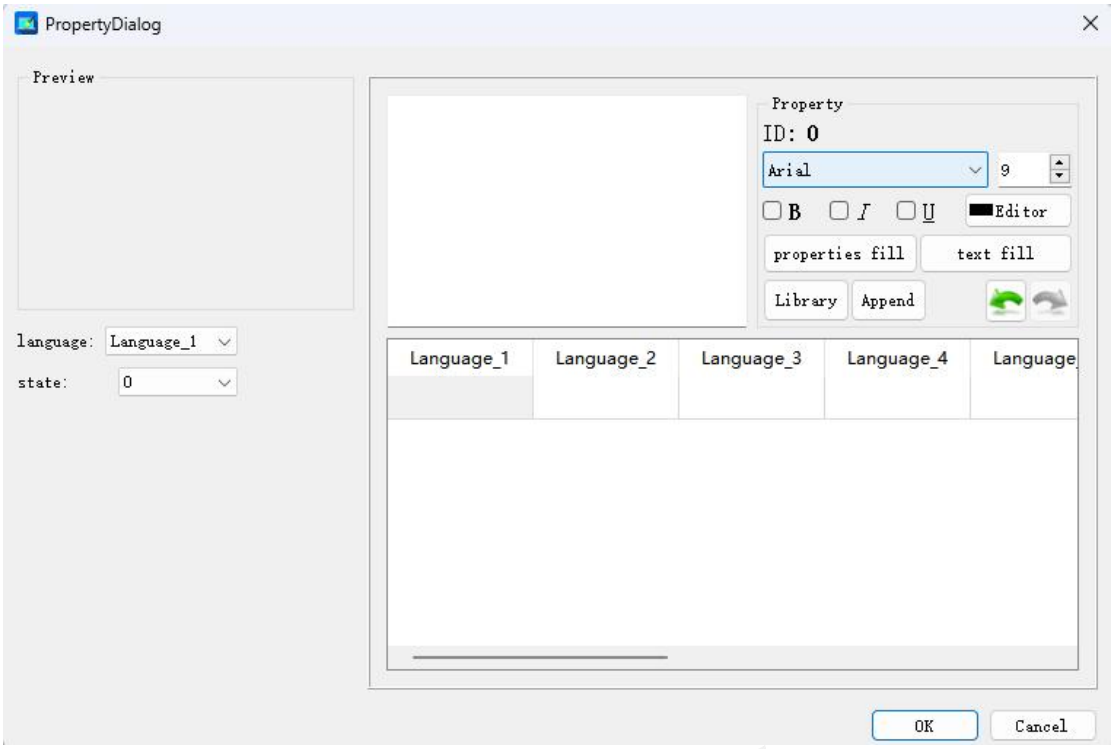
[Display Title]

Indicates whether the component's title is visible.

[Title]




This indicates setting the title of the component, with a button available for multi-language settings.

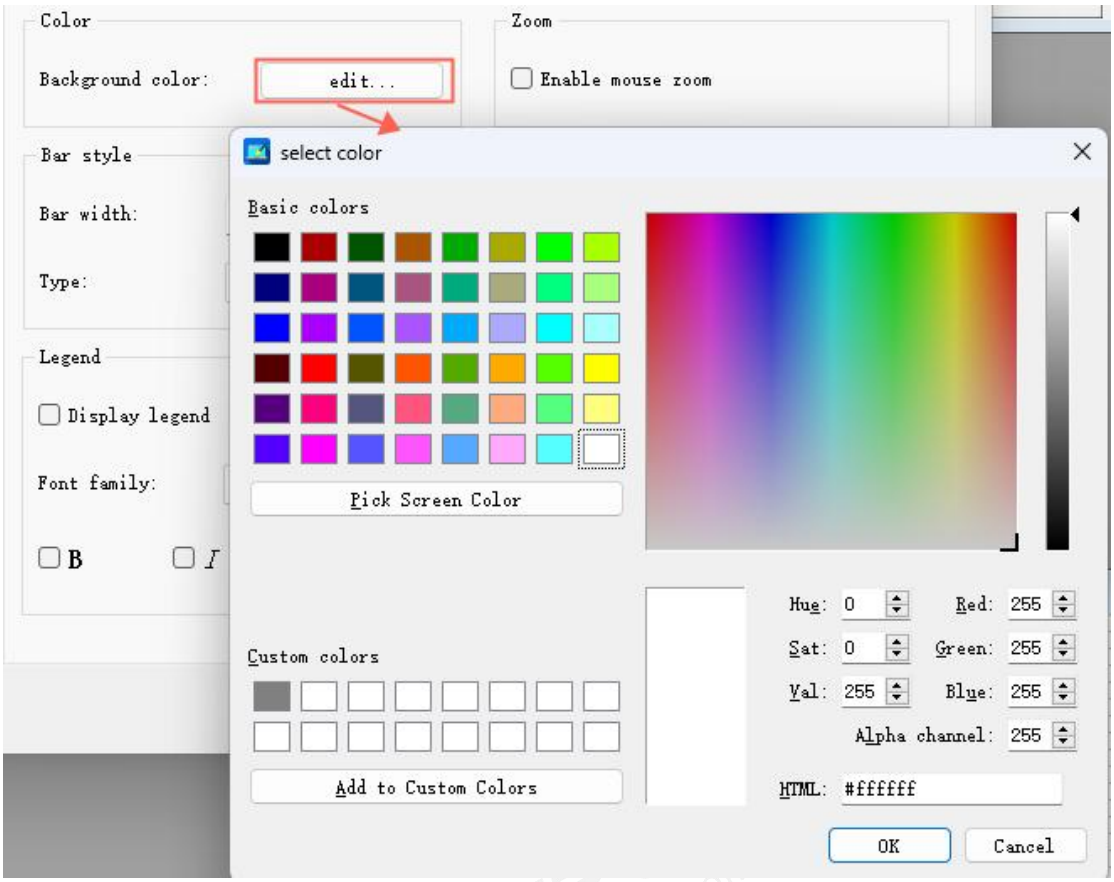



[Font Family]

Set the font family for the component title.

[Color]

Background color




This indicates setting the component's background color by clicking the


button.

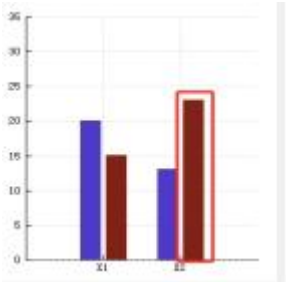
[Zoom]

**Enable mouse zoom**

Set whether the component has mouse zoom functionality.

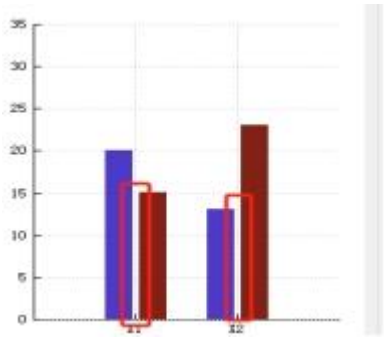
[Bar Style]

**Bar width**



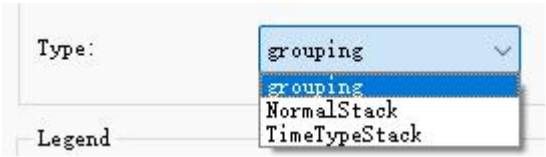
Used to set the width of the bars.

### Series spacing



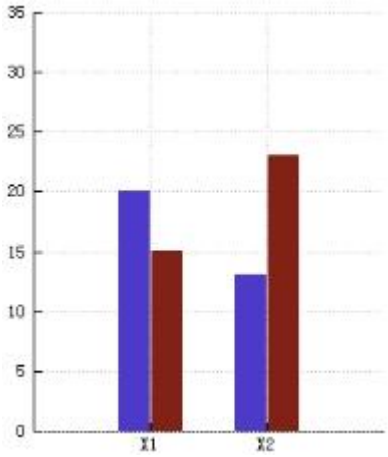
Used to set the display spacing between data series.

### Chart type




Used to set the chart type for the component.

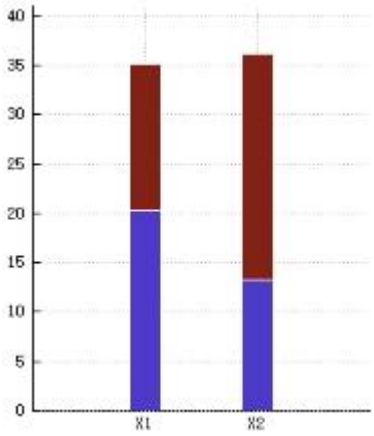
### Grouping




Note: This type of chart requires the following data structure format:

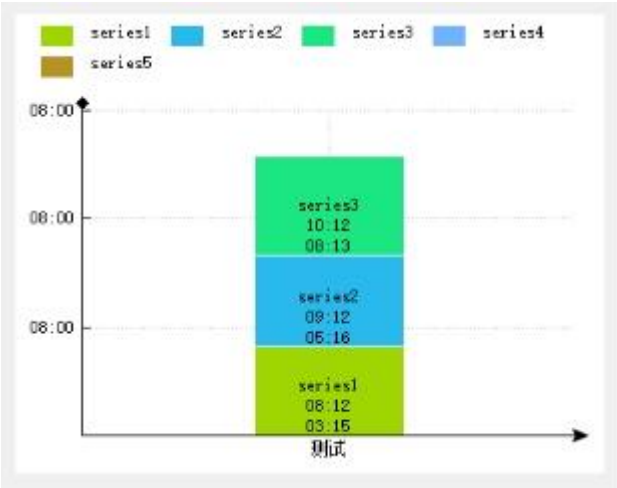
```
[
  {
    "s":1,
    "x":"12/03",
    "y":"20"
  },
  {
    "s":1,
    "x":"12/04",
    "y":"50"
  },
  {
    "s":2,
    "x":"12/03",
    "y":"40"
```

**Normal stacking**




Note: This type of chart requires the same data structure format as "Grouping".

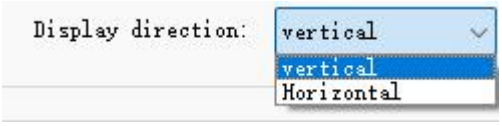
**Time-based stacking**



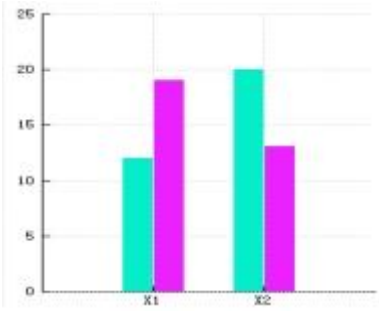

Note: This type of chart requires the following data structure format:

```
[
  {
    "x":"测试",
    "y":{
      "startTime":"2022-09-18 08:12:12",
      "endTime":"2022-09-19 03:15:11"
    },
    "s":"1"
  },
  {
    "x":"测试",
    "y":{
      "startTime":"2022-09-19 09:12:12",
      "endTime":"2022-09-20 05:16:17"
    },
    "s":"2"
  },
```

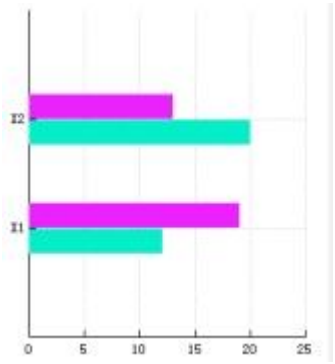
**Display orientation**




■ Vertical




■ Horizontal

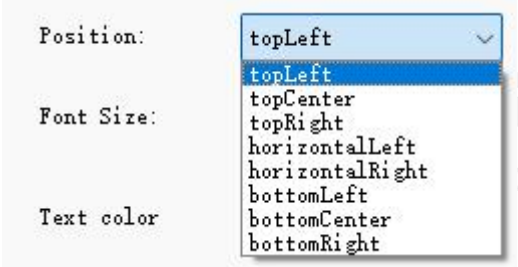



[Legend]

**Display**

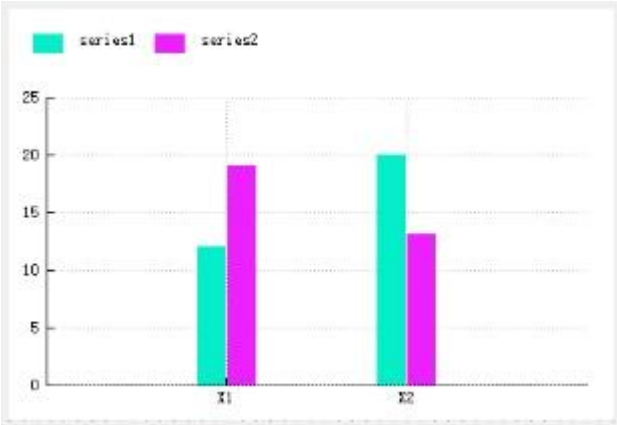
Indicates whether the component's legend is visible.

**Position**

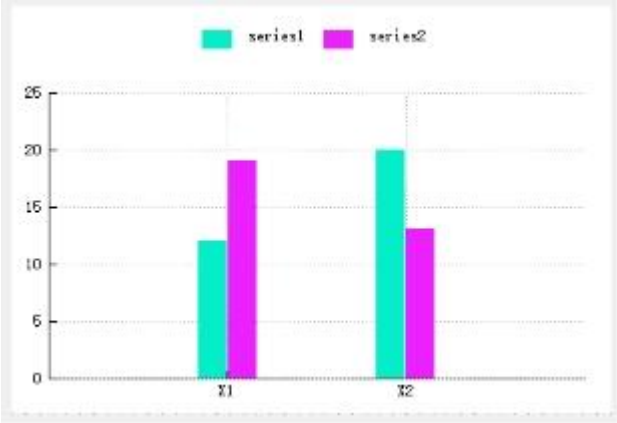



Specifies the display position of the legend.

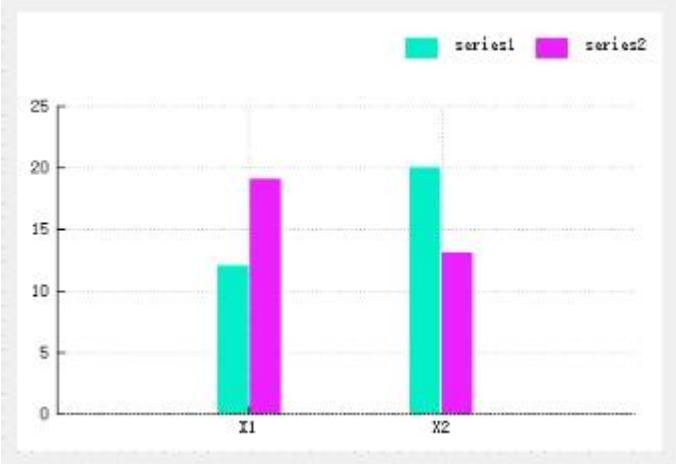
Top Left



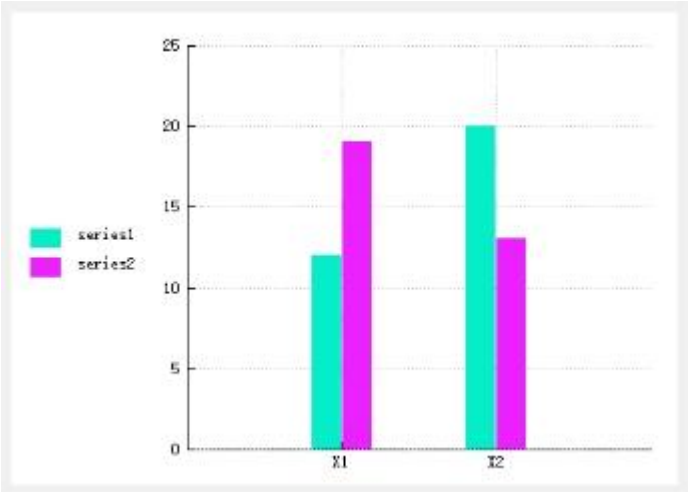

Top Center



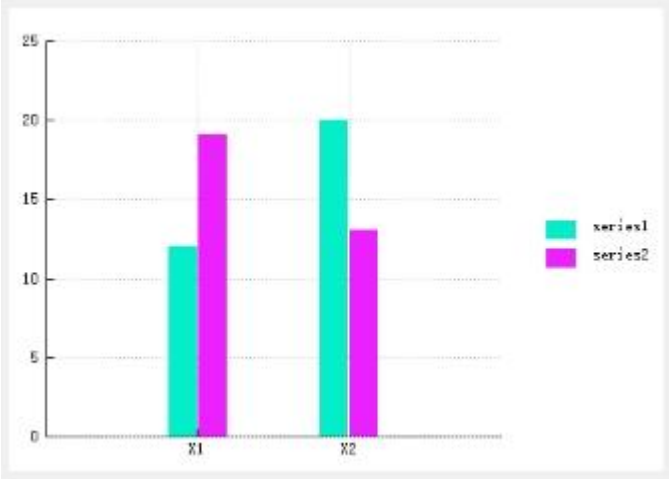

Top Right



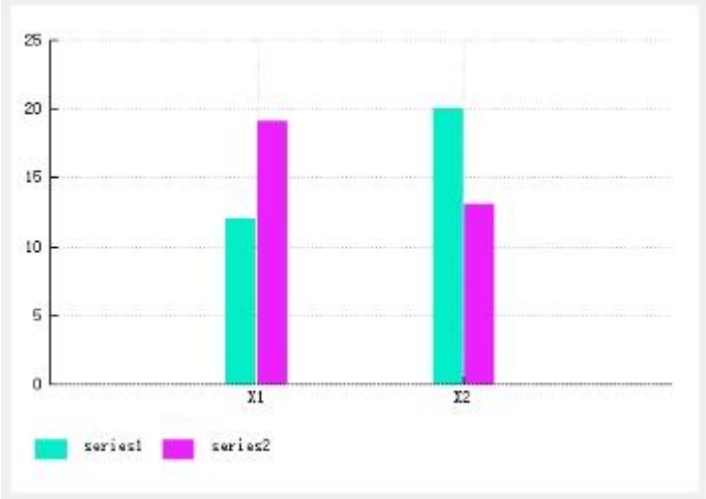

Horizontal Left



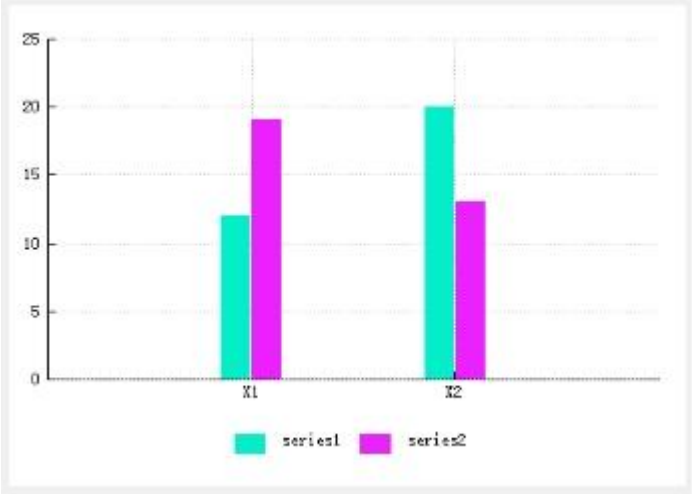

Horizontal Right



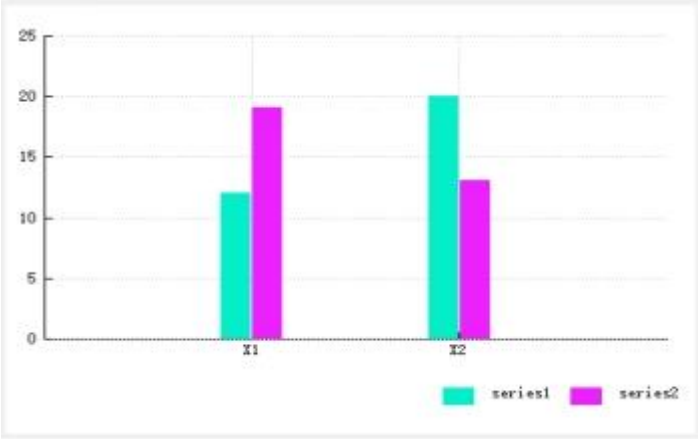

Bottom Left




Bottom Center



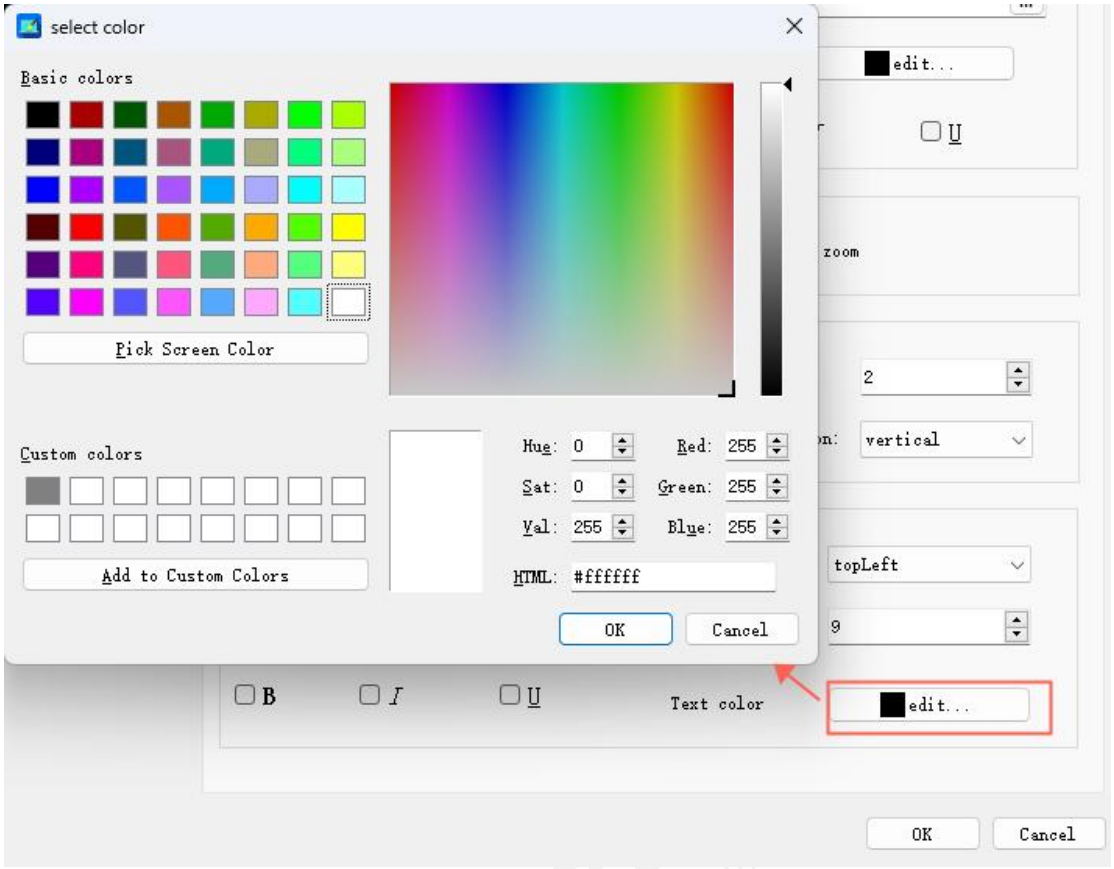

Bottom Right

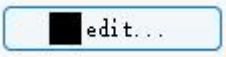



**Font Family**

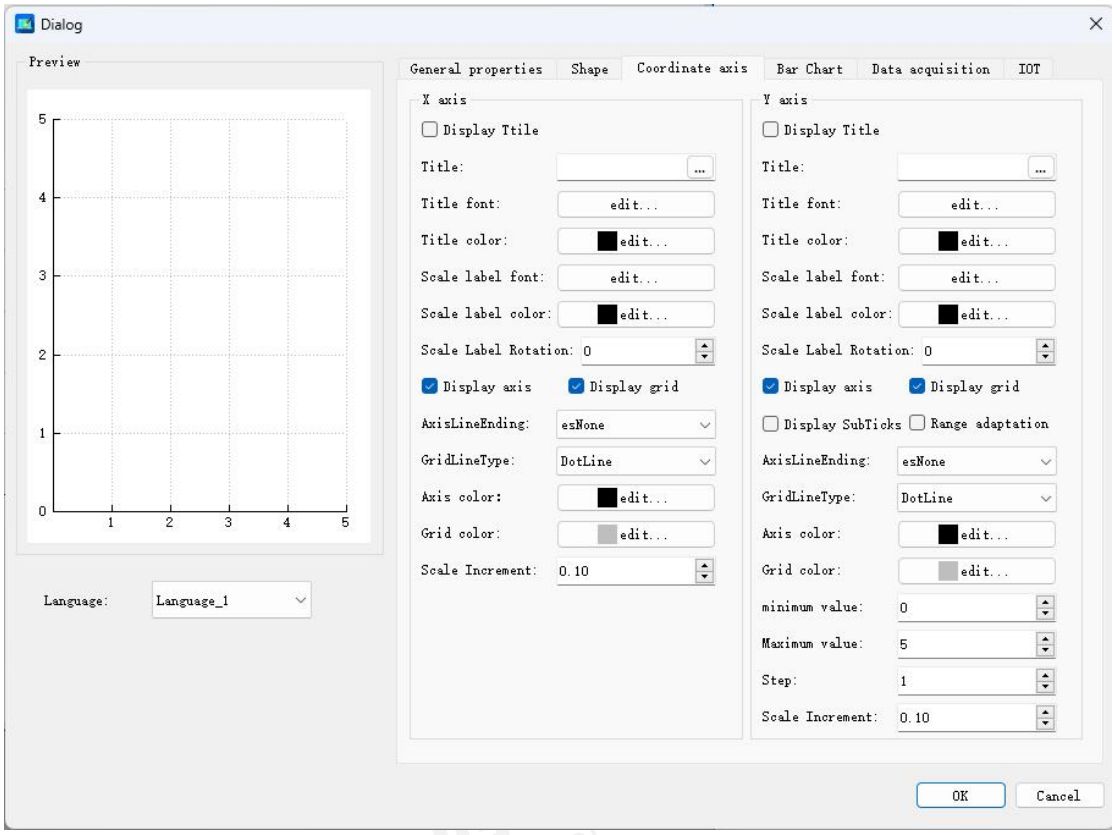
Set the font family for the legend text.

Text Color



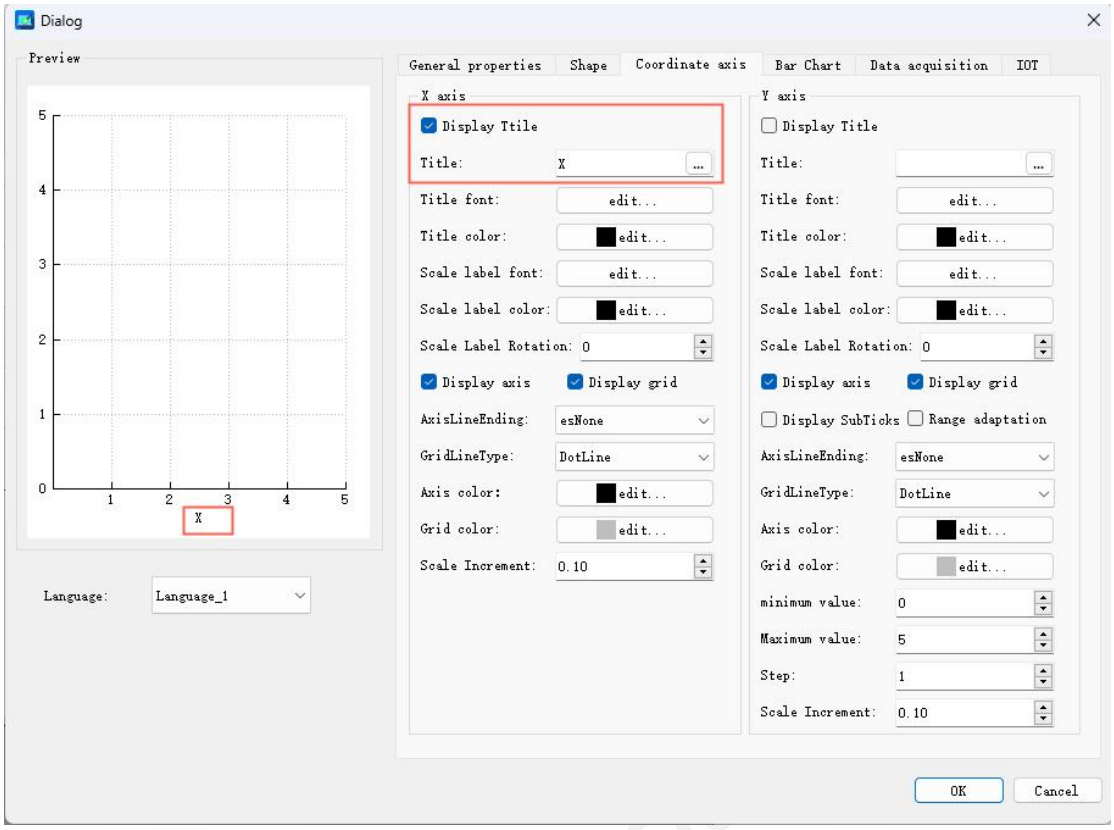

This indicates setting the font color for the legend text of the component by clicking the  button.

## 6.21.3. Axes




[X Axis]

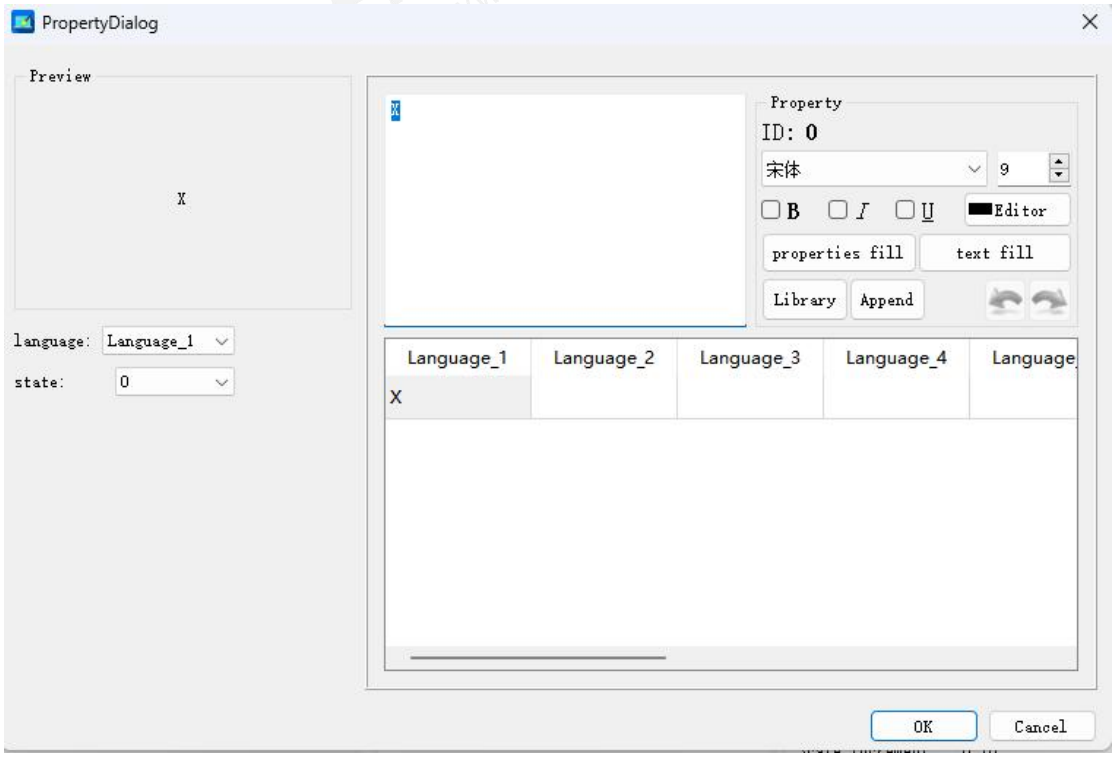
[Display Title]



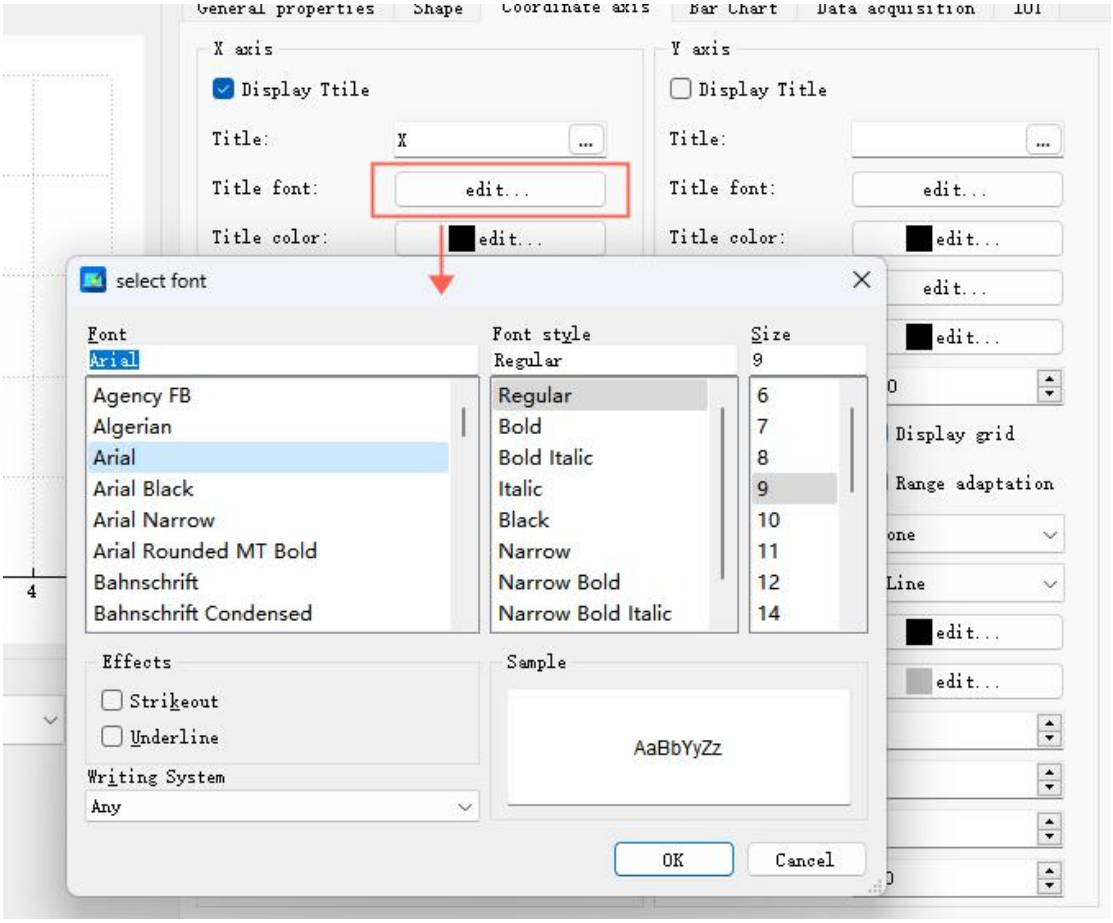

Display X-axis title.

[Title]

Set the title for the X-axis. Click the button for multilingual settings.

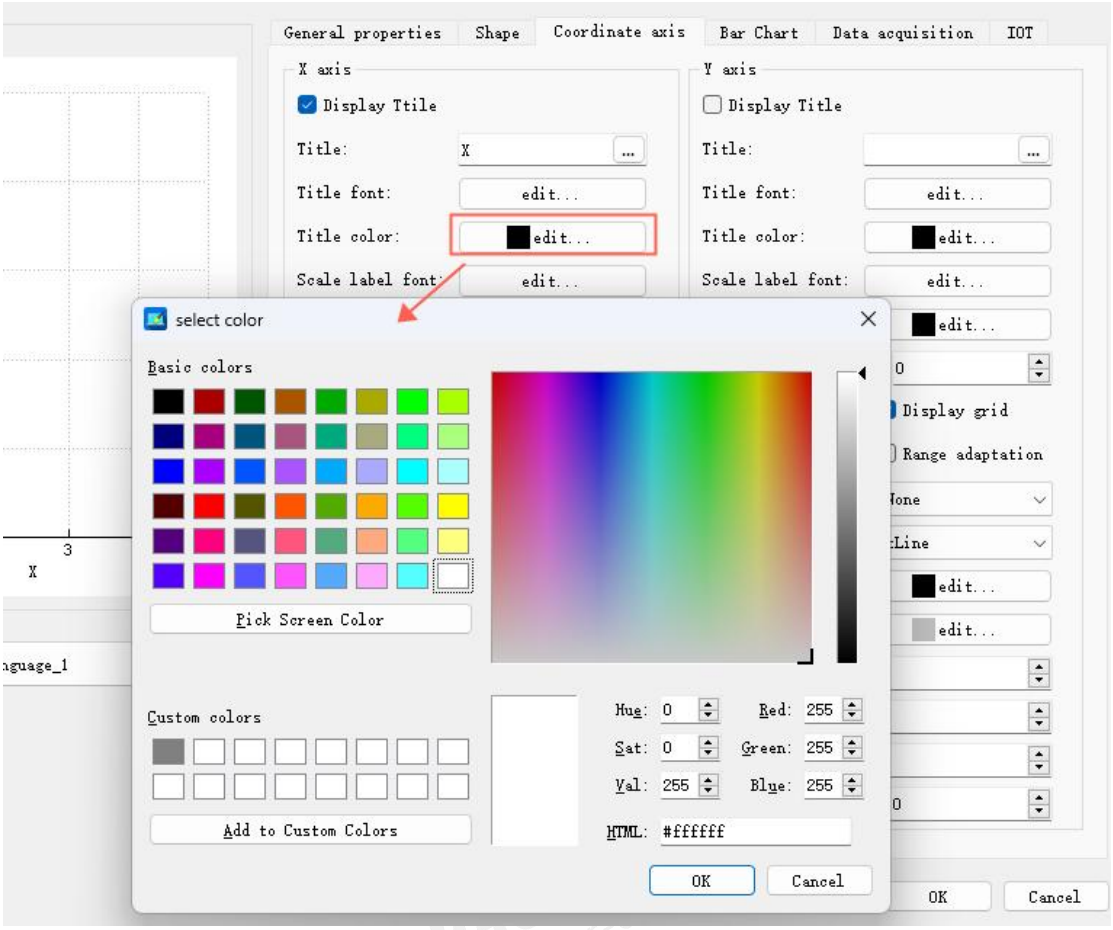


[Title font]

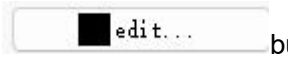



Sets the font for the X-axis title.

[Title color]

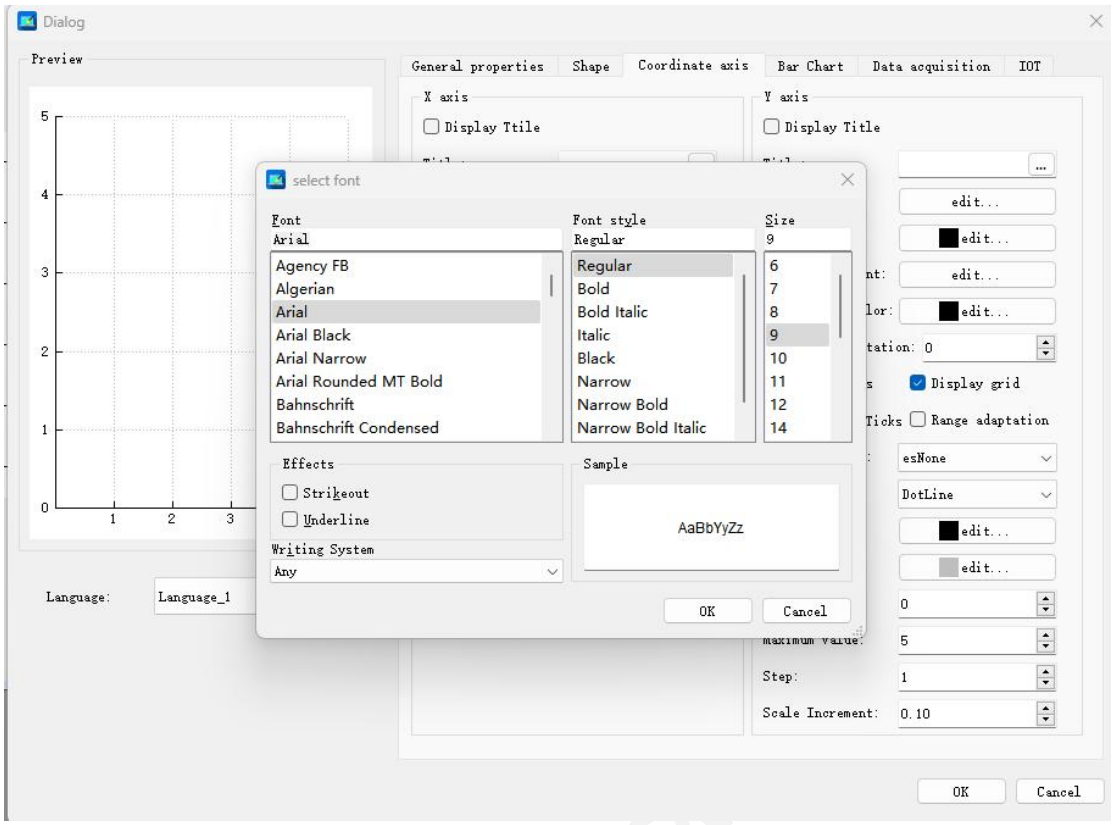



Indicates that the font color of the component's X-axis title text can be set by clicking the



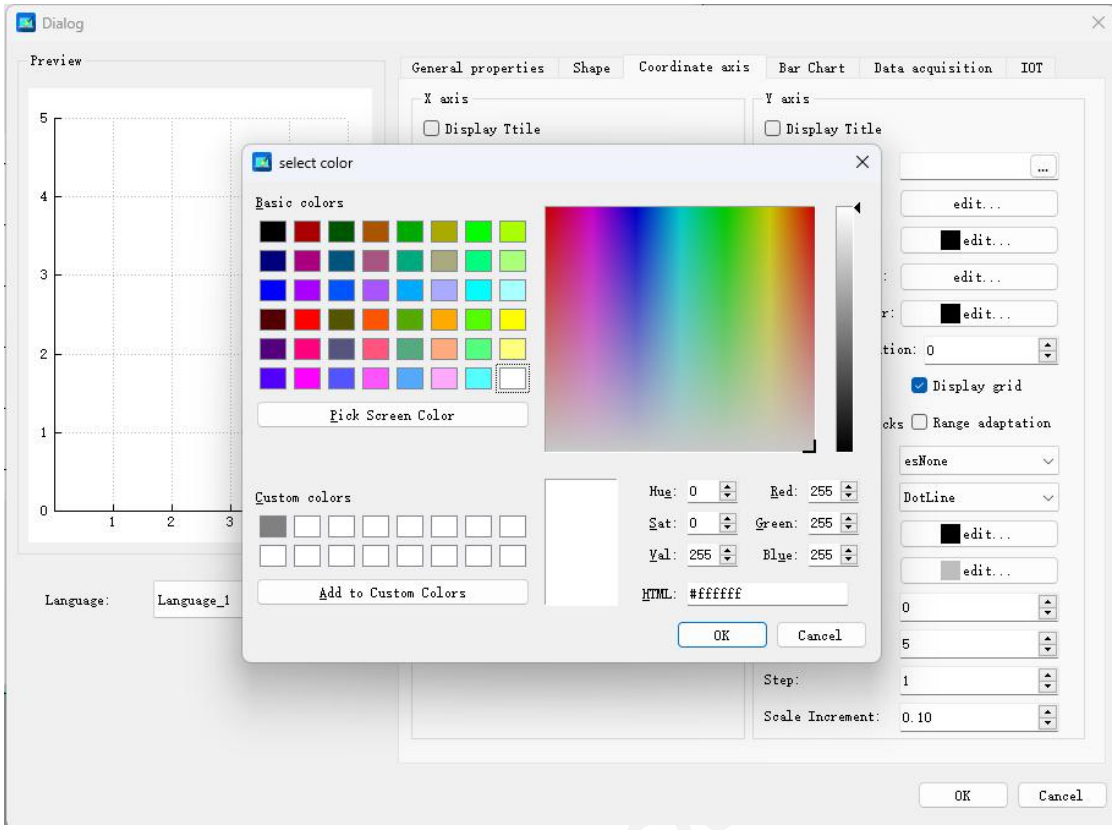

button.

[Scale Label Fonts]




Sets the font for the X-axis scale labels.

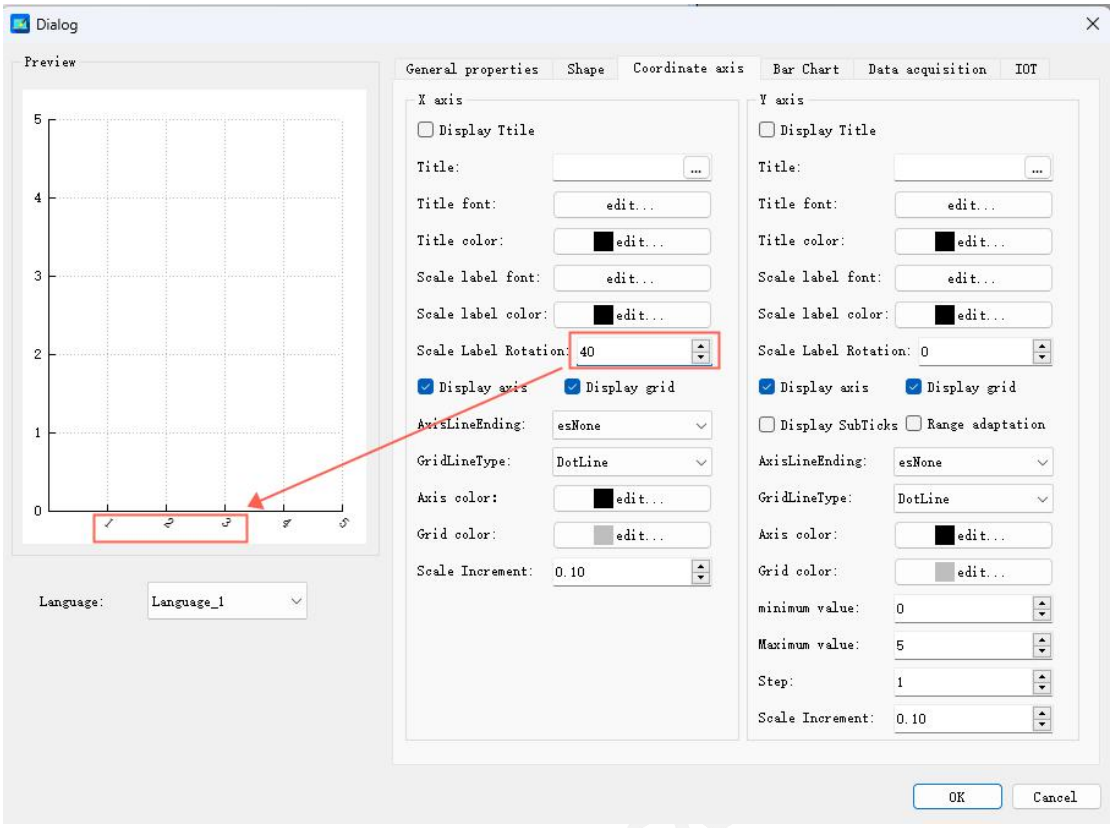
[Scale Label Color]




Indicates that the font color of the component's X-axis scale label can be set by clicking the  button.



[Scale Label Tilt]




Indicates to set the tilt angle of the X-axis scale label. The range is 0-180.

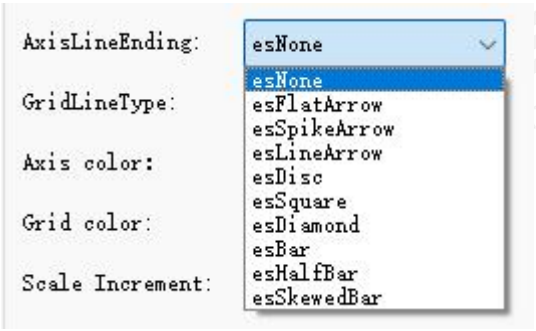
[Display Axis Lines]

Sets whether or not to display the X-axis axis lines.

[Display Grid Lines

Used to set whether the X-axis gridlines are displayed.

[Axis Endpoints]




Used to set the axial end style of the X-axis.

**esNone**

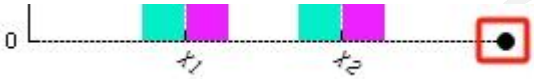
Indicates that there is no endpoint style.

**esFlatArrow**


**esSpikeArrow**


**esLineArrow**


**esDisc**




**esSquare**

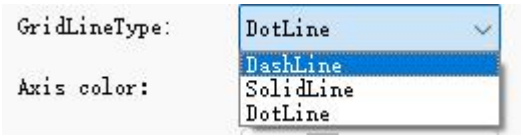

**esDiamond**


**esBar**


**esHalfBar**

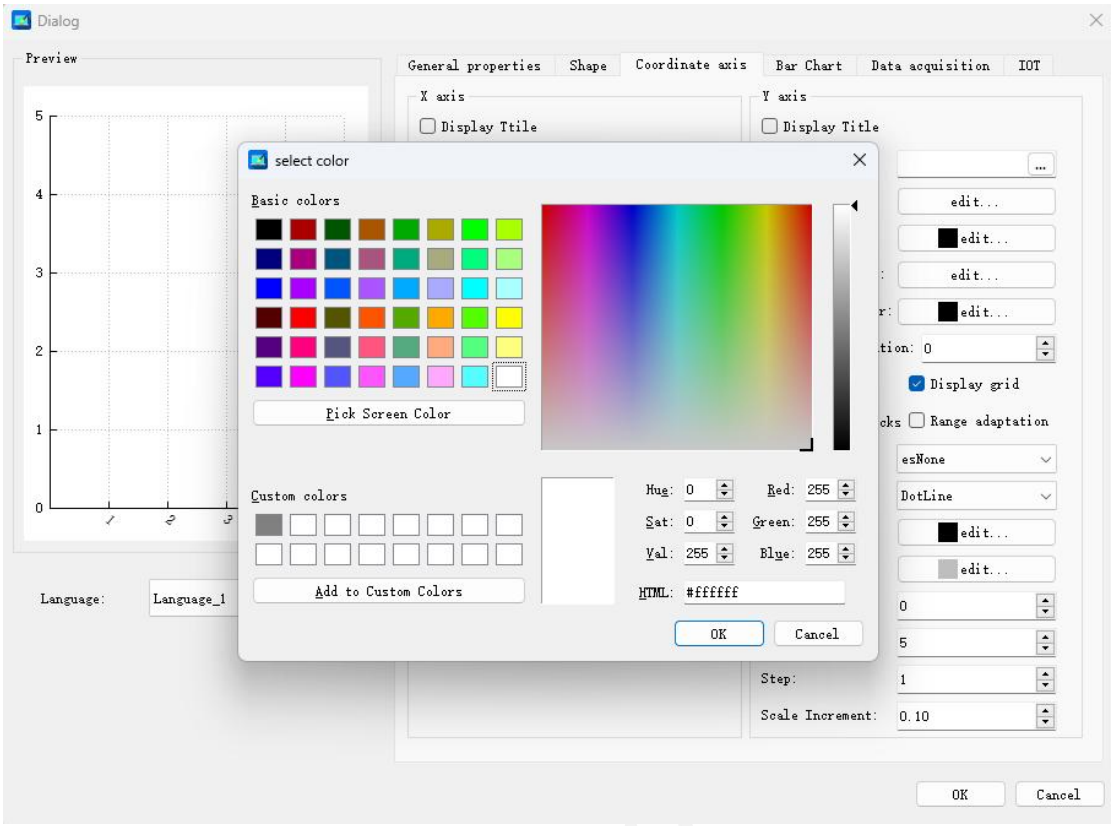

**esSkewedBar**

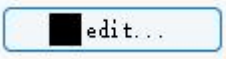

[Grid line pattern]




Indicates that the grid line pattern for the X-axis is set.

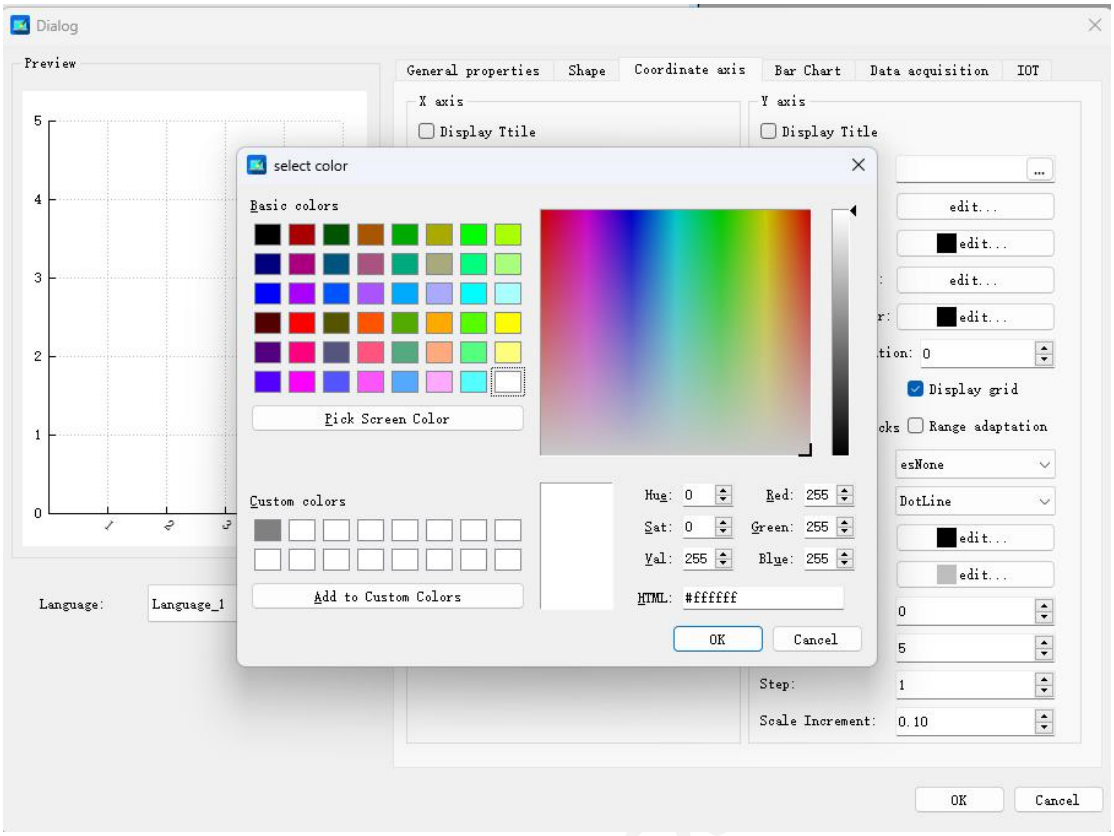
[Axis color]




Indicates that the color of the component's X-axis axis can be set by clicking the 

button.

[Grid line color]




Indicates that the color of the component's X-axis gridlines can be set by clicking the 

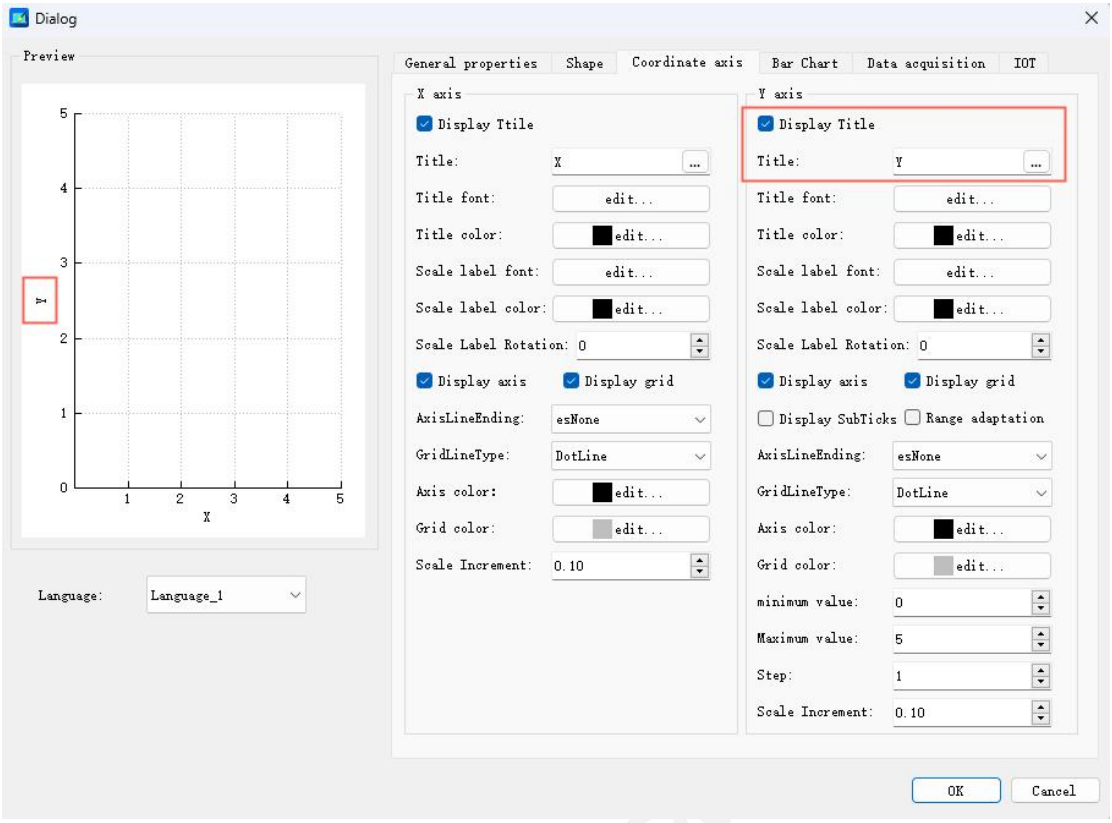
button.

Zoom Increment

Indicates the zoom increment for the X-axis when the "Mouse Zoom" function is enabled.

[Y-axis]

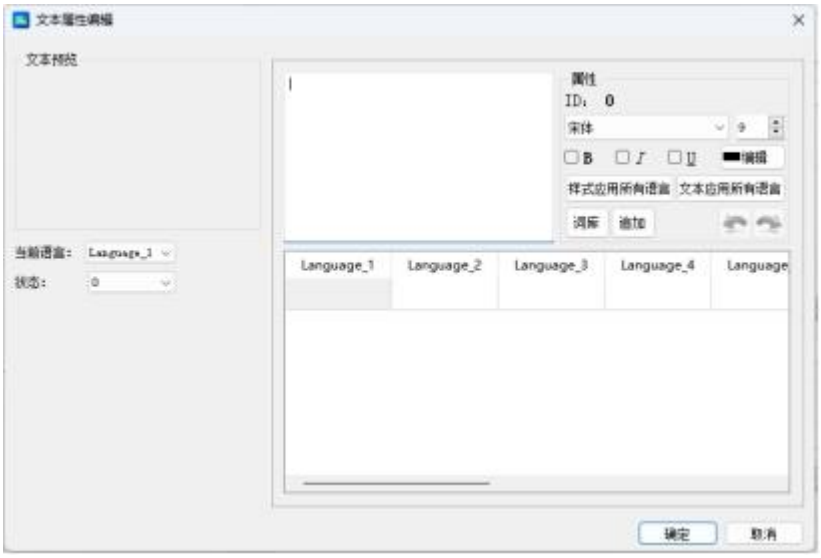
[Display Title]



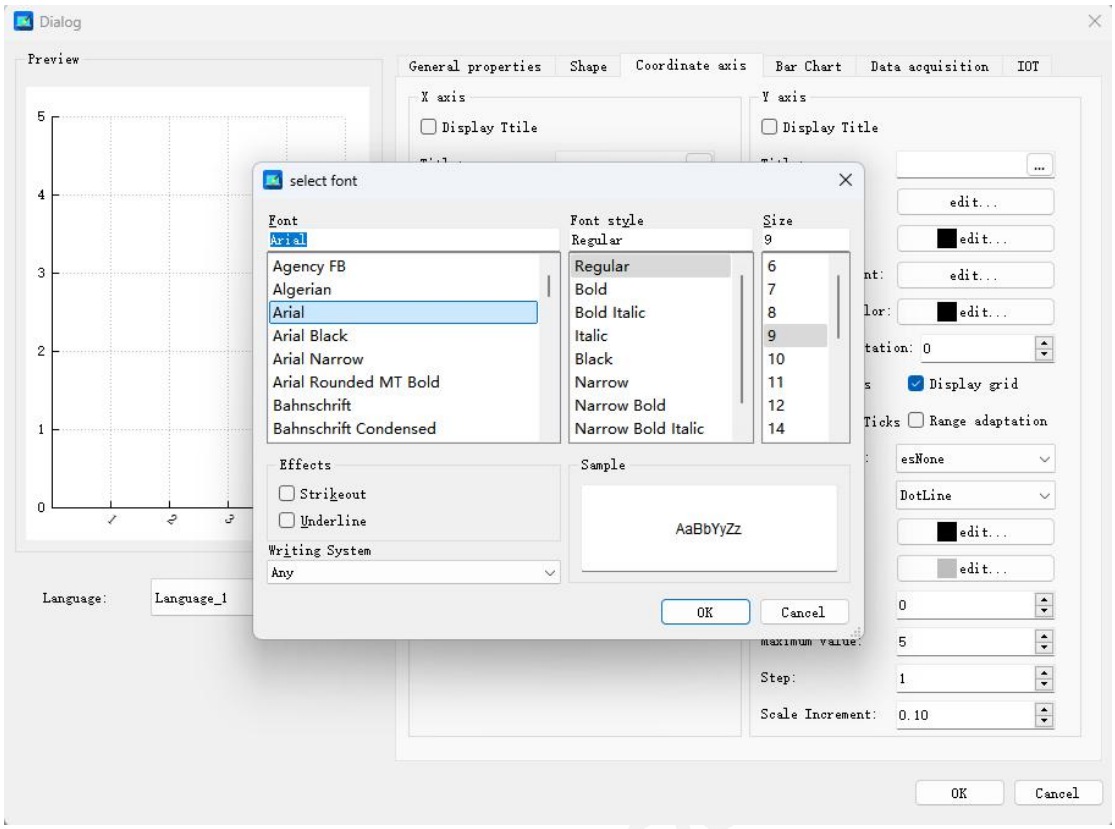

Sets whether to display the Y-axis title.

[Title]

Sets the title of the Y-axis. Click the ... button to make multi-language settings.

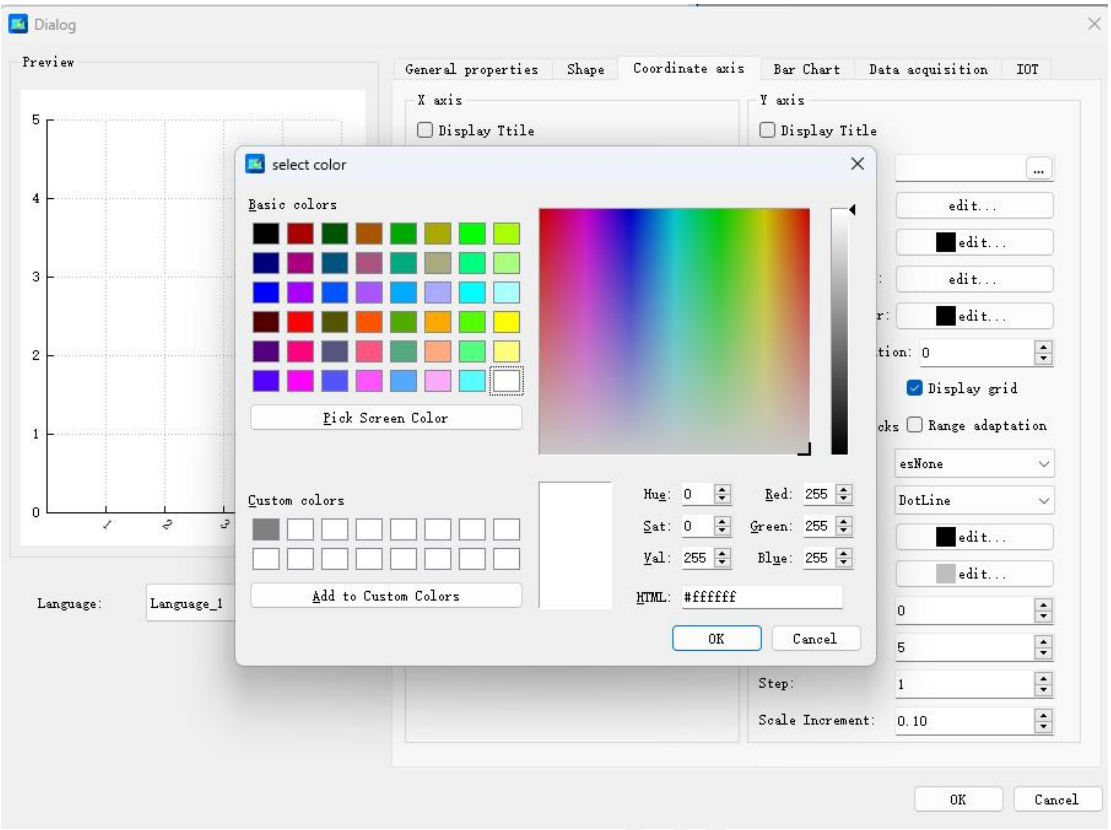



[Title font]




Sets the font for the Y-axis title.

[Title color]




Indicates that the font color of the component's Y-axis title text can be set by clicking the 

button.

[Scale Label Font]

Sets the font for the Y-axis title.

[Scale Label Color]

Sets the color of the Y-axis scale label.

[Scale Label Tilt]

Indicates that the tilt angle of the Y-axis scale label is set. The range is 0-180.

[Display Axis]

Indicates whether or not to display the Y-axis axes.

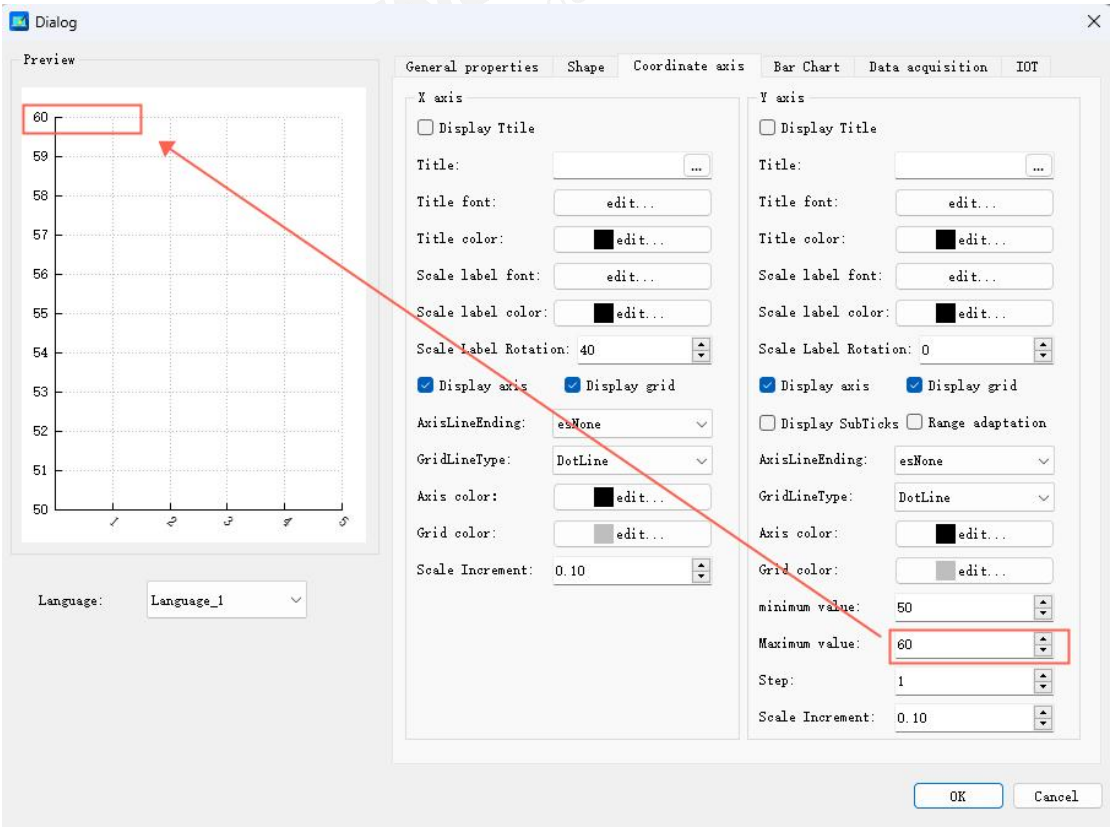
[Display Grid Lines]

Sets whether or not to display the Y-axis grid lines.

[Display sub-scale]

Set whether to display the Y-axis subscale.

Value Range Adaptation




Sets whether or not the value range is adaptive.

[Axis End]

Set the axial endpoint style of the Y-axis, referring to the above "X-axis >> Axial Endpoint" description above.

[Grid Line Style]

Sets the line style of the Y-axis grid lines.

[Axis Line Color]

Sets the color of the Y-axis line.

[Grid Line Color]

Sets the color of the Y-axis grid lines.

[Minimum Value]

Sets the minimum value of the Y-axis scale. This setting is displayed under the premise that the "Appearance >> Bar Style >> Chart Type" is selected as non-"Stacked by Time Type".

[Maximum Value]

Sets the maximum value of the Y-axis scale. This setting is displayed under the premise that the "Appearance >> Bar Style >> Chart Type" is selected as non-"Stacked by Time Type".

[Label Time Format]

Sets the display format of Y-axis labels when they are in time format. This setting is displayed under the premise that "Appearance >> Bar Style >> Chart Type" is selected as "Stacked by Time Type".

[Data Time Format]

Sets the specified format for the time data received by the component when the chart type is set to "Stacked by Time Type". This setting is displayed under the premise that "Appearance >> Bar Style >> Chart Type" is not selected as "Stacked by Time Type".

For example, when using the macro interface "setIOTData (set bar chart data)" to set data, if you want the startTime and endTime fields of the data below to be accepted by the component, you need to set this item to yyyy-MM-dd HH:mm:ss:

```
[
  {
    "x":"测试",
    "y":{"startTime": "2022-09-18 08:12:12","endTime": "2022-09-19 03:15:11"},
    "s":"1"
  },
  {
    "x":"测试",
    "y":{"startTime": "2022-09-19 09:12:12","endTime": "2022-09-20 05:16:17"},
    "s":"2"
  },
  {
    "x":"测试",
    "y":{"startTime": "2022-09-20 10:12:12","endTime": "2022-09-21 08:13:16"},
    "s":"3"
  }
];
```

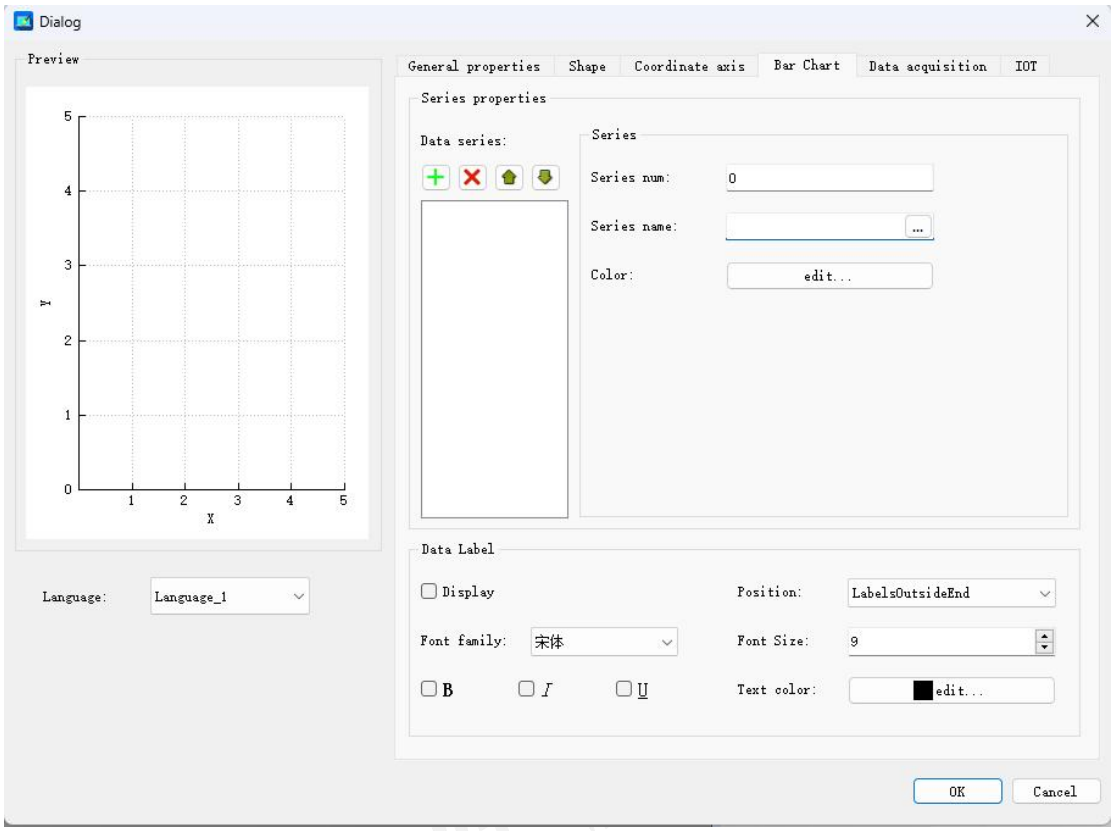
[Step Size]

Set the step size of the Y-axis

[Zoom increments]

The zoom increment represents the scaling increment of the Y-axis when the "mouse zoom" function is enabled.

# 6.21.4. Bar Charts




[Data Series]

**Additional**


Cl

icking  on the button will add a new data series.

**Delete**

Clicking  the button will delete a data series.

**Move Up**

Clicking  the button will move the selected data series up.

**Move Down**

Clicking  the button will move the selected data series down.

[Series]

**Series Number**

Displays the number of the data series, not editable.

**Series Name**

Used to set the name of the data series. Click  the button for multilingual settings.

**Series Color**

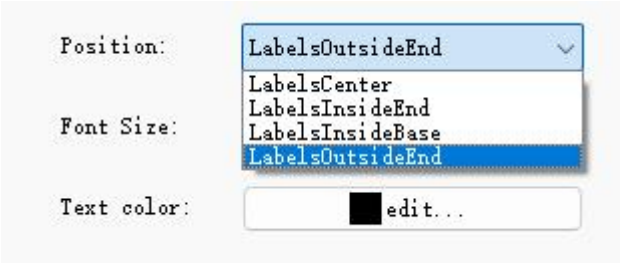
Used to set the color of the data series.

[Data Labels]

**Display**

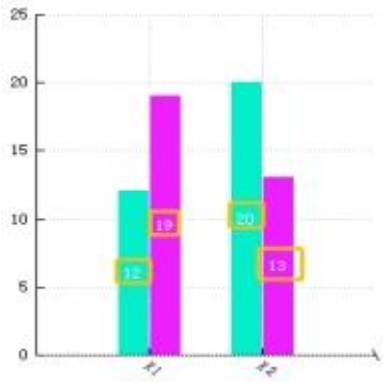
Used to set whether to display data labels for the bars.

**Placement**

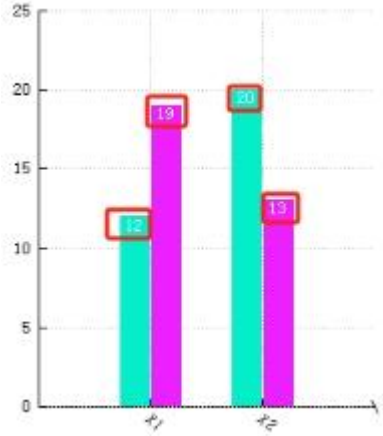



Indicates to set the display position of the data label relative to the bar.

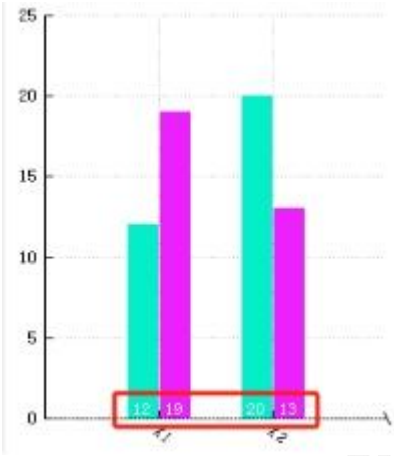
**LabelsCenter**



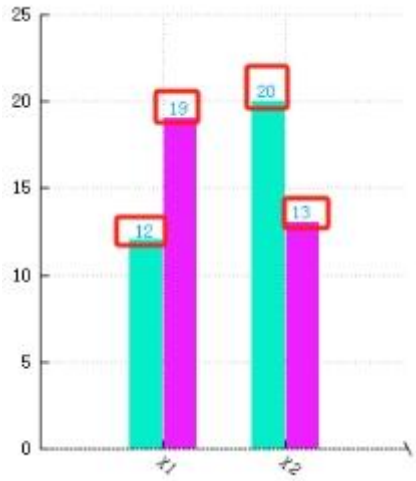

**LabelsInsideEnd**




**LabelsInsideBase**



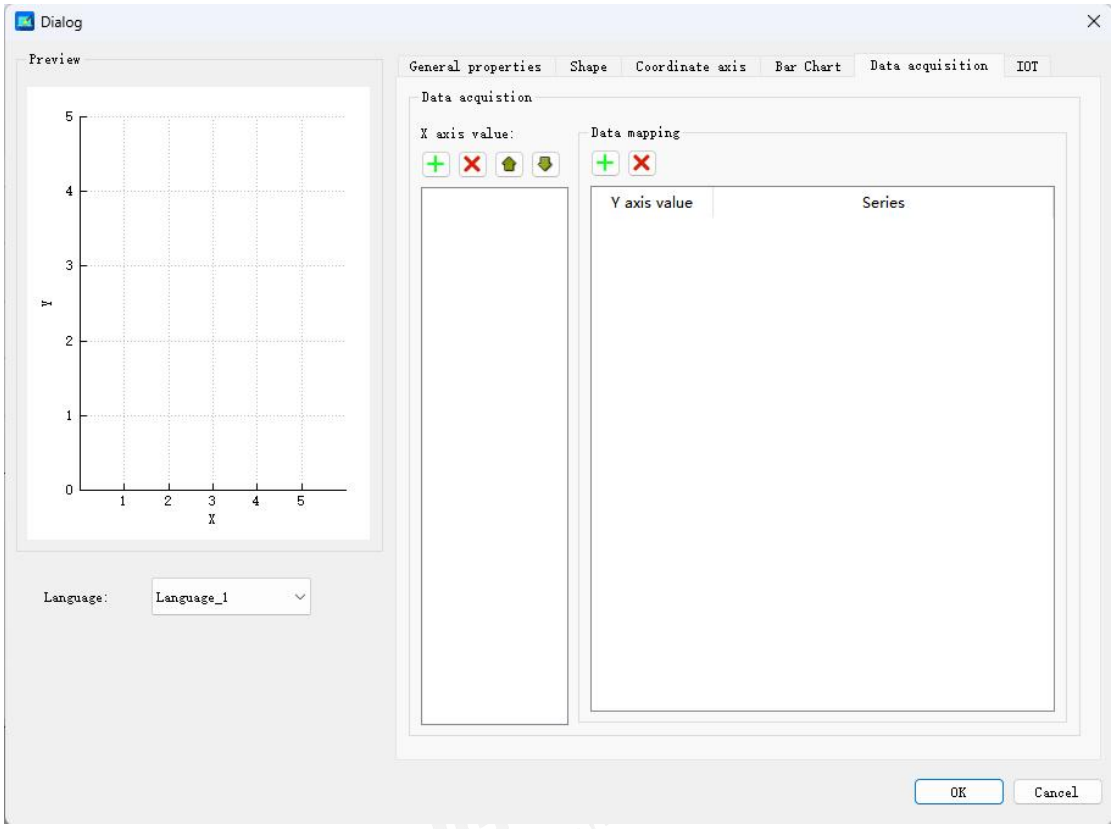

**LabelsOutsideEnd**




[Font Family]

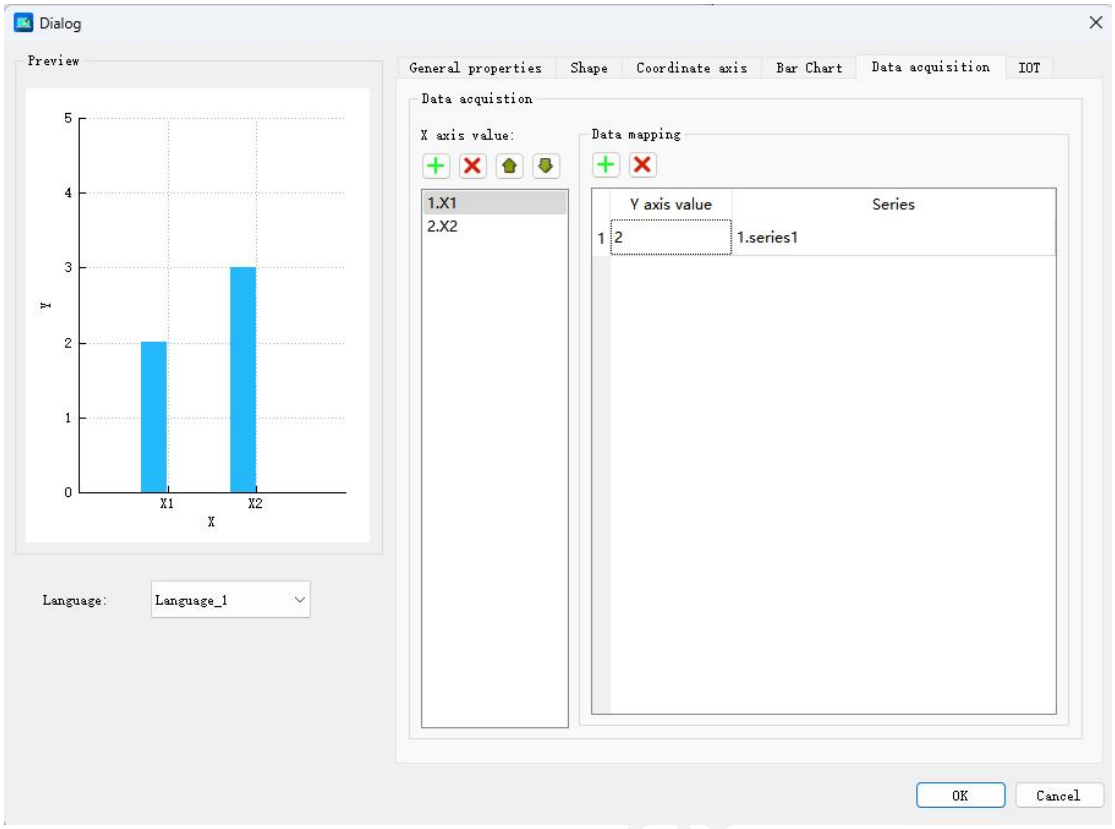
Used to set the font family for data labels.

# 6.21.5. Data Acquisition




[X Axis Text Labels]

**Add**




Click the button to add a new X-axis text label.

**Delete**

Click the button to delete an X-axis text label.

**Move Up**

Click the button to move the selected X-axis text label up.

**Move Down**

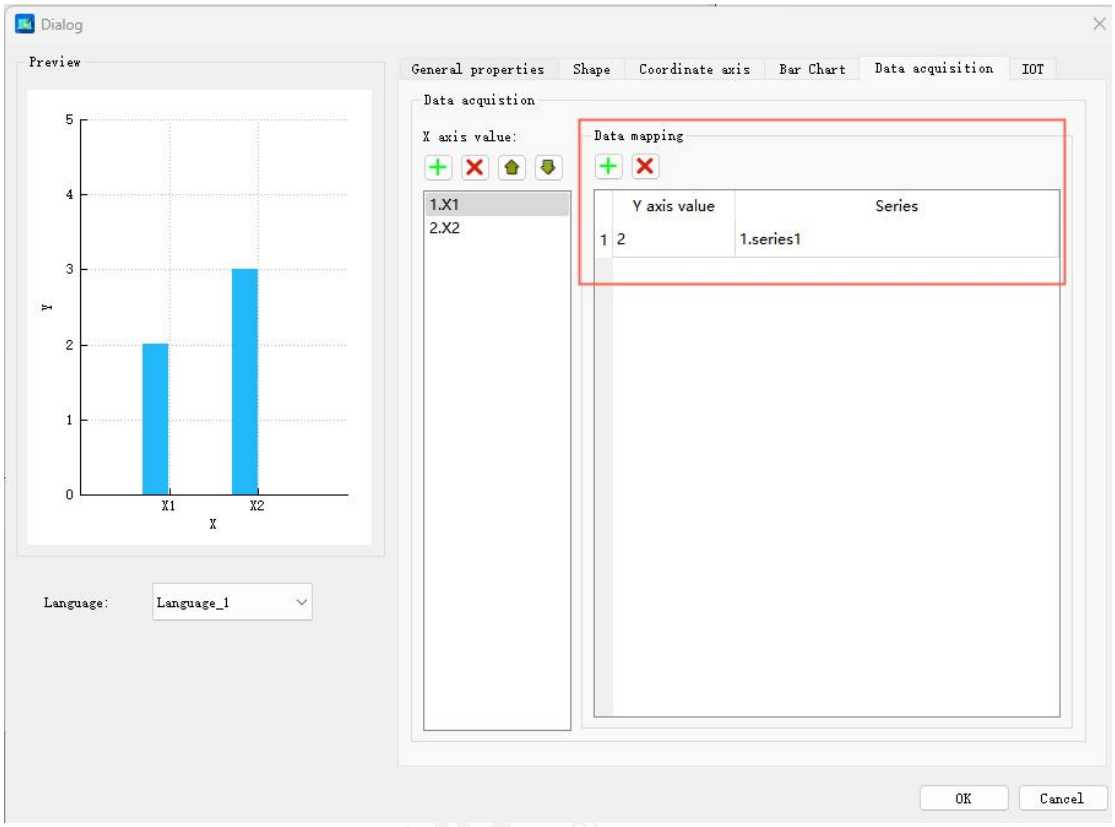
Click the button to move the selected X-axis text label down.

**Edit X-axis text label**

You can switch the language first, then double-click the "X Axis Text Label" to edit the text.

### [Data Mapping]

### Add



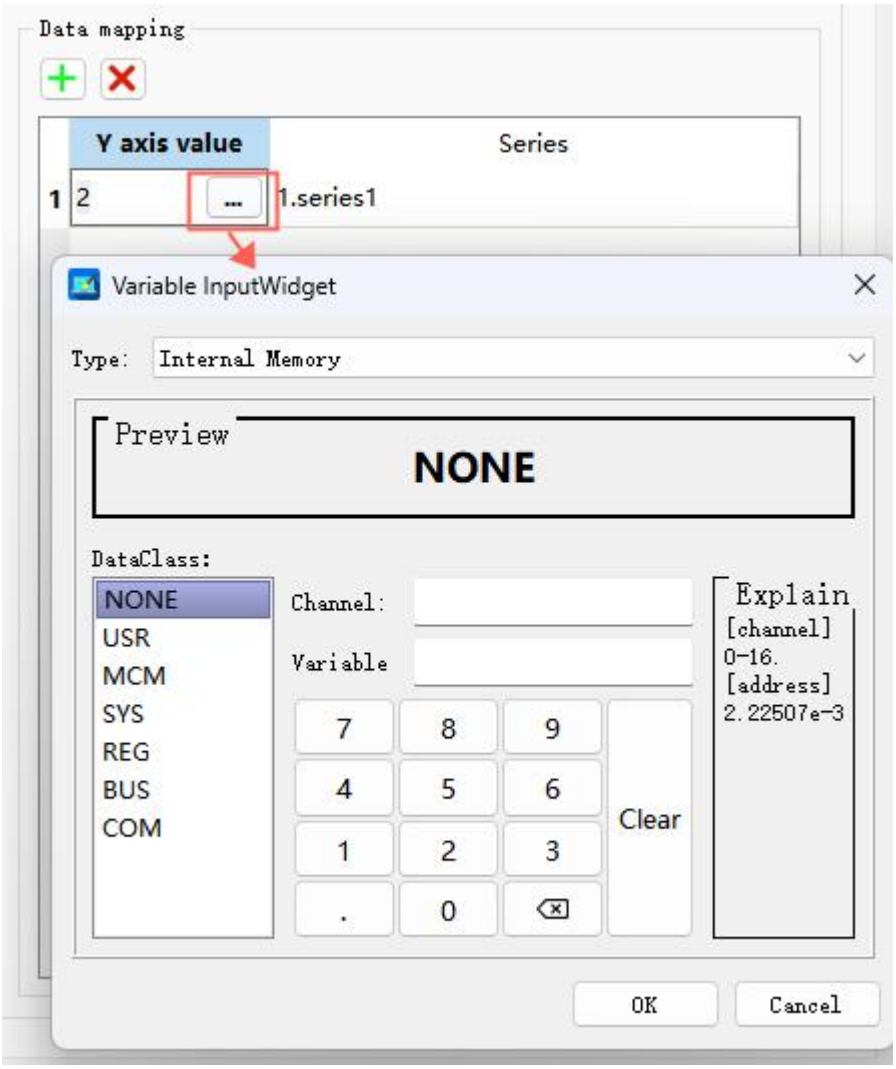

Click the button to add a new data mapping.

### Delete

Click the button to delete a data mapping.

### Table Data Description


### Y-axis Value




Double-click the cell to directly enter a value as static data.

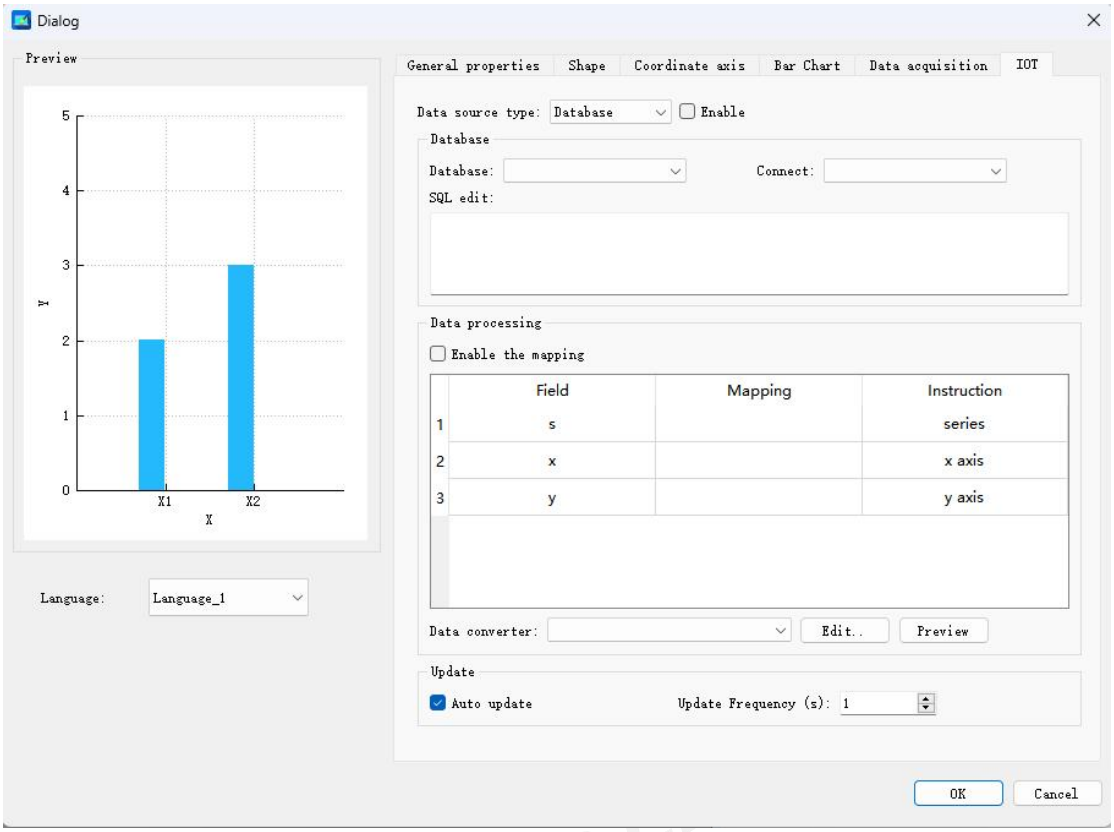
Or click the ... button to open the variable editing window to associate a variable address.

Note: Currently, variable settings are not supported (this is reserved for future implementation).

**Series**


Double-click the cell and select the data series from the dropdown list.

## 6.21.6. IoT




This section is mainly used to set properties related to the Internet of Things (IoT).

[Data Source Type]

**Data Source Type**

Used to set the data source for IoT.

**Enable**

Indicates whether to enable IoT functionality.

[Database]

**Database**

Select the type of database to use from the dropdown list. Currently, only MySQL is supported. This setting is displayed only when "Database" is selected as the "Data Source Type".

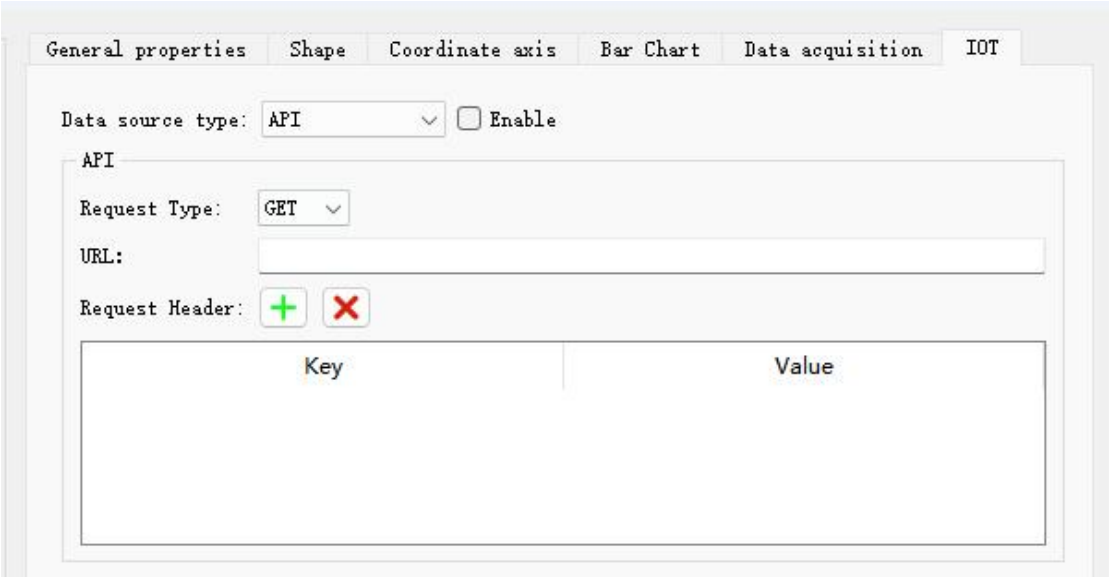
**Connection**

Select the database connection from the dropdown list. The dropdown list items are sourced from the database connection configurations in the IoT settings. This setting is displayed only when "Database" is selected as the "Data Source Type".

**SQL Editor**

Used to edit SQL query statements, for example: select col, col2 from table. This setting is displayed only when "Database" is selected as the "Data Source Type".

**API**



**Request Method**

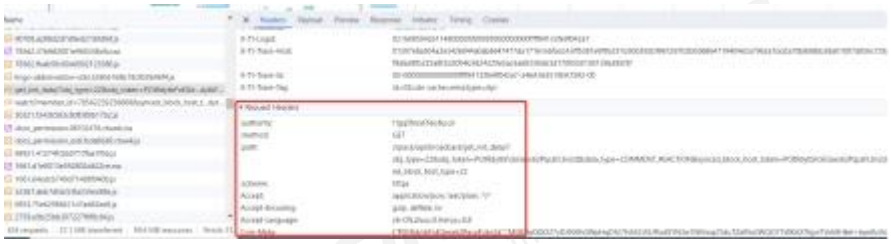

Used to set the request method for API requests. This setting is displayed only when "API" is selected as the "Data Source Type".

**URL**

Used to set the API request URL, for example: https://www.example.com. This setting is displayed only when "API" is selected as the "Data Source Type".

**Request Headers**

Used to set the headers for API requests in key-value pairs, such as:



Click the  button to add, and click the  button to delete.

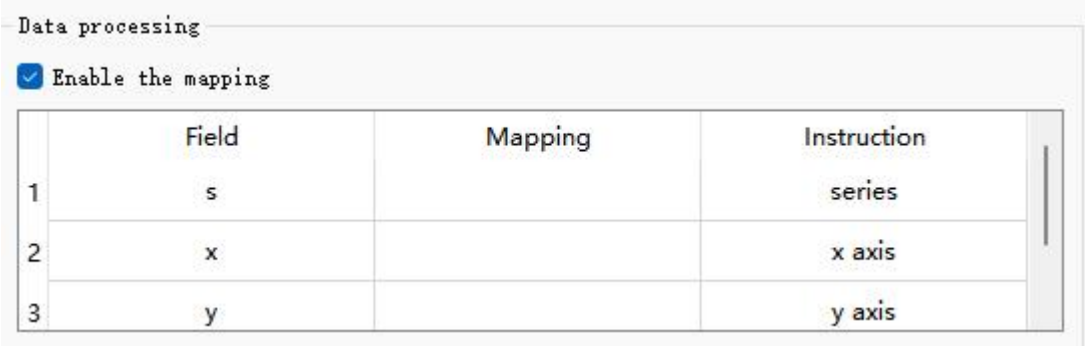
These settings are displayed only when "API" is selected as the "Data Source Type".

[Data Processing]

**Enable Mapping**

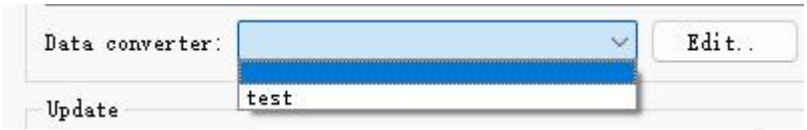
Set whether to enable data field mapping. If mapping is not enabled, the component's data requires fields to be s, x, y (refer to the mapping table below). Otherwise, the data will not be successfully set in the component.

**Mapping Table**



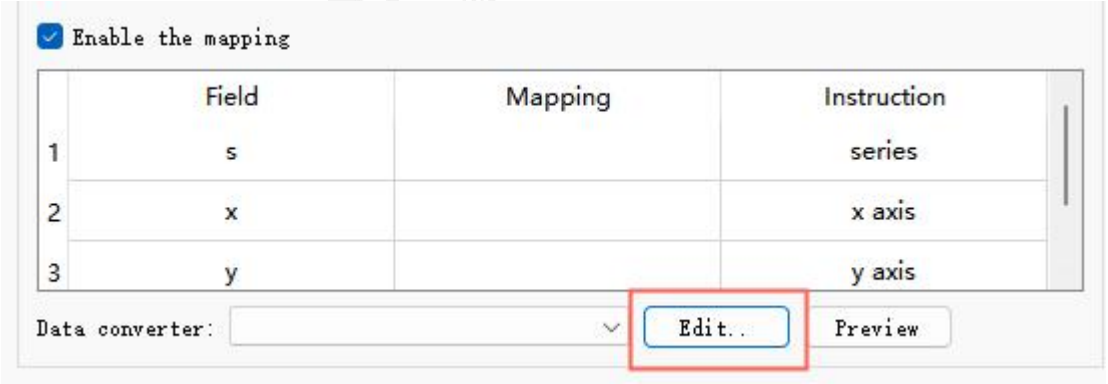

When mapping is enabled, if the data fields retrieved from the database query are not s, x, y, users can map the retrieved data fields to the required s, x, y fields of the component.

**Data Converter**




Select the data converter to use from the dropdown list. The dropdown list items are sourced from the data converters created in the "Edit" section below.

**Edit**




Manage data converters by clicking the 

button as shown above.

**Preview**

Open the data preview window by clicking the 

button as shown above.

[Update]

**Auto-update**

Indicates whether the component is enabled to automatically fetch IoT data at regular intervals for refreshing.

**Update frequency**

Specifies the interval in seconds (s) for the automatic refresh of component data.

## 6.21.7. Component Macro Interface

**1. Enabled Zoom（Enable zoom functionality）**

```
/**
* @brief Enable or disable zoom functionality.
* @param enabled - Enable status [true: enabled; false: disabled]
* @return void
*/
void enabledZoom(const bool &enabled);
```

Example：Form.hBarChart.enabledZoom(true)

**2. Export Content（Export IoT data）**

```
/**
* @description Export data (supports only .csv, .xlsx, or .xls formats)
* @param {number} type - Export type [0: export data; 1: export graph; 2: export data and graph]
* @param {string} path - Save path (full path including file name and extension)
* @returns {void}
*/
void exportContent(const int type, QString path = "");
```

Example：Form.hBarChart.exportContent(0,"C:\\Users\\123.csv")

Note: This functionality only works on the PCBased platform.

### 3. setAxisZoom（Set axis scaling）

```
/**
* @description Set axis scaling.
* @param {number} direction - Scaling direction [0: move down; 1: move up; 2: move left; 3: move right; 4: shrink; 5: zoom in; 6: fit]
* @returns {void}
*/
void setAxisZoom(const int &direction);
```

Example：Form.hBarChart.setAxisZoom(0)

### 4. setAxisZoomIncrement（Setting axis scaling increments）

```
/**
* @description Set axis scaling increment.
* @param {number} axis - Axis [0: X axis; 1: Y axis]
* @param {number} increment - Scaling increment
* @returns {void}
*/
void setAxisZoomIncrement(const int &axis, const double &increment);
```

Example：Form.hBarChart.setAxisZoomIncrement(0,10.00)

### 5. setBarChartBackground（Setting the background color of a bar chart）

```
/**
* @description Set bar chart background color.
* @param {string} color - Color name
* @returns {void}
*/
void setBarChartBackground(const QString &color);
```

Example:Form.hBarChart.setBarChartBackground("#b7ffd4")

### 6. setBarChartLegend（Setting up a Bar Chart Legend）

```
/**
* @description Set bar chart legend.
* @param {number} position - Legend position [0: top-left; 1: top-center; 2: top-right; 3: horizontal-left; 4: horizontal-right; 5: bottom-left; 6: bottom-center; 7: bottom-right]
* @param {string} family - Font family
* @param {number} fontSize - Font size
* @param {boolean} bold - Whether bold
* @param {boolean} italic - Whether italic
* @param {boolean} underline - Whether underline
* @param {string} color - Font color
* @returns {void}
*/
void setBarChartLegend(const int position = HBarChart::topCenter, const QString &family = "Microsoft YaHei", const int fontSize = 9, const bool bold = false, const bool italic = false, const bool underline = false, const QString &color = "#000000");
```

Example：Form.hBarChart.setBarChartLegend(1,"Microsoft YaHei",true,true,true,"#000000")

### 7. setBarChartSeries（Setting up Bar Chart Data Series）

```
/**
* @description Set bar chart data series.
* @param {string} series - JSON string of data series
* [{"id":"1","name":"series1","seriesColor":"#ffffff"},...]
* @returns {void}
*/
void setBarChartSeries(const QString &series);
```

Example：

Form.hBarChart.setBarChartSeries("[{\"id\":\"1\",\"name\":\"series1\",\"seriesColor\":\"#f737ca\"},{\"id\":\"2\",\"name\":\"series2\",\"seriesColor\":\"#b7ffd4\"}]")

### 8. setBarChartTitle（Setting the Bar Chart Title）

```
/**
* @description Set bar chart title (calling this method makes the title visible by default).
* @param {string} title - Title
* @param {string} family - Font family
* @param {number} fontSize - Font size
* @param {boolean} bold - Whether bold
* @param {boolean} italic - Whether italic
* @param {boolean} underline - Whether underline
* @param {string} color - Font color
* @returns {void}
*/
void setBarChartTitle(const QString &title, const QString &family = "Microsoft YaHei", const int fontSize = 9, const bool bold = false, const bool italic = false, const bool underline = false, const QString &color = "#000000");
```

Example ： Form.hBarChart.setBarChartTitle("Title 1","Microsoft YaHei",true,true,true,"#000000")

### 9. setBarChartXAxisColor（Setting the Column X axis line color）

```
/**
* @description Set X-axis line color.
* @param {string} color - Color
* @returns {void}
*/
void setBarChartXAxisColor(const QString &color);
```

Example：Form.hBarChart.setBarChartXAxisColor("#000000")

### 10. setBarChartXGridColor（Setting the Column X-axis grid line color）

```
/**
* @description Set X-axis grid line color.
* @param {string} color - Color
* @returns {void}
*/
void setBarChartXGridColor(const QString &color);
```

Example：Form.hBarChart.setBarChartXGridColor("#000000")

### 11. setBarChartXGridLineType（Setting the bar chart X-axis grid line pattern）

```
/**
* @description Set X-axis grid line style.
* @param {number} type - Line style [0: dashed; 1: solid; 2: dotted]
* @returns {void}*/
void setBarChartXGridLineType(const int type);
```

Example：Form.hBarChart.setBarChartXGridLineType(1)

### 12. setBarChartXTitle（Setting the bar chart X-axis title）

```
/**
* @description Set bar chart X-axis title (calling this method makes the title visible by default).
* @param {string} title - Title
* @param {string} family - Font family
* @param {number} fontSize - Font size
* @param {boolean} bold - Whether bold
* @param {boolean} italic - Whether italic
* @param {boolean} underline - Whether underline
*   @param {string} color - Font color
* @returns {void}*/
void setBarChartXTitle(const QString &title, const QString &family = "Microsoft YaHei", const int fontSize = 9, const bool bold = false, const bool italic = false, const bool underline = false, const QString &color = "#000000");
```

Example：Form.hBarChart.setBarChartXTitle("title 1","Microsoft YaHei",true,true,true,"#000000")

### 13. setBarChartXUpperEnding（Setting the bar X axis axial endpoint style）

```
/**
* @brief Set X-axis axis endpoint style.
* @param {number} style - Style [0: esNone; 1: esFlatArrow; 2: esSpikeArrow; 3: esLineArrow; 4: esDisc; 5: esSquare; 6: esDiamond; 7: esBar; 8: esHalfBar; 9: esSkewedBar]
* @returns {void}
*/
void setBarChartXUpperEnding(const int style);
```

Example：Form.hBarChart.setBarChartXUpperEnding(1)

**14. setBarChartYAxisColor（Setting the bar Y-axis line color）**

```
/**
* @description Set Y-axis line color.
* @param {string} color - Color
* @returns {void}
*/
void setBarChartYAxisColor(const QString &color);
```

Example：Form.hBarChart.setBarChartYAxisColor("#000000")

**15. setBarChartYAxisValRangeAutoAdjust （Setting the bar Y-axis value range adaptation）**

```
 /**
* @description Set bar chart Y-axis auto-fit data range.
* @param {boolean} enabled - Whether to enable auto-fit
* @returns {void}
*/
void setBarChartYAxisValRangeAutoAdjust(const bool enabled = false);
```

Example：Form.hBarChart.setBarChartYAxisValRangeAutoAdjust(true)

**16. setBarChartYGridColor（Setting the bar Y-axis grid line color）**

```
/**
* @description Set Y-axis grid line color.
* @param {string} color - Color
* @returns {void}
*/
void setBarChartYGridColor(const QString &color);
```

Example：Form.hBarChart.setBarChartYGridColor("#000000")

### 17. setBarChartYGridLineType（Setting the histogram Y-axis grid line pattern）

```
/**
* @description Set Y-axis grid line style.
* @param {number} type - Line style [0: dashed; 1: solid; 2: dotted]
* @returns {void} */
*/
void setBarChartYGridLineType(const int type);
```

Example：Form.hBarChart.setBarChartYGridLineType(1)

### 18. setBarChartYRange（Setting the range of bar Y-axis values）

```
/**
* @Description Set the range of values on the Y-axis for the bar chart.
* @Param minimum The minimum value
* @Param maximum The maximum value
* @Return None*/
void setBarChartYRange(const int mininum = 0, const int maxinum =
10);
```

Example：Form.hBarChart.setBarChartYRange(0,20)

### 19. setBarChartYStep（Setting the Histogram Y-axis Scale Steps）

```
/**
* @Description Set the step size of the Y-axis scale for the bar chart.
* @Param step The step size
* @Return None*/
void setBarChartYStep(const int step = 1);
```

Example：Form.hBarChart.setBarChartYStep(2)

### 20. setBarChartYTitle（Setting the bar Y-axis title）

```
/**
* @Description Set the Y-axis title for the bar chart (calling this method makes the title visible by default).
* @Param title The title
* @Param family Font family
* @Param fontSize Font size
* @Param bold Whether it's bold
* @Param italic Whether it's italicized
* @Param underline Whether it's underlined
* @Param color Font color
* @Return None*/
void setBarChartYTitle(const QString &title, const QString &family = "Microsoft YaHei", const int fontSize = 9, const bool bold = false, const bool italic = false, const bool underline = false, const QString &color = "#000000");
```

Example ： Form.hBarChart.setBarChartYTitle(“title1","Microsoft YaHei",true,true,true,"#000000")

### 21. setBarChartYUpperEnding（Setting the bar Y-axis axial endpoint style）

```
/**
* @Description Set the Y-axis end point style.
* @Param style Style [0: esNone; 1: esFlatArrow; 2: esSpikeArrow; 3: esLineArrow; 4: esDisc; 5: esSquare; 6: esDiamond; 7: esBar; 8: esHalfBar; 9: esSkewedBar]
* @Return None*/
void setBarChartYUpperEnding(const int style);
```

Example：Form.hBarChart.setBarChartYUpperEnding(1)

### 22. setBarsStyle（Setting the bar bar style）

```
/**
* @Description Set the bar style for the bar chart.
* @Param direction Direction of display [0: vertical; 1: horizontal]
* @Param barWidth Bar width
* @Param spacing Series spacing
* @Param xLabelsRotation Rotation angle of X-axis text labels
* @Return None*/
void setBarsStyle(const int direction = 0, const int barWidth = 20, const int spacing = 2, const int xLabelsRotation = 0);
```

Example:Form.hBarChart.setBarsStyle(0,30,5,0)

### 23. setIOTData（Setting the bar chart data）

```
/**
* @Description Set data for a time-series stacked chart.
* @Param data JSON string data [{"x":"2022/09/18 08:12:12","y":"2022/09/19 03:15:11","s":"1"},...]
* @Return None*/
  void setIOTData(QString data);
```

Example:Form.hBarChart.setIOTData("[{\"x\":\"2022/09/18 08:12:12\",\"y\":\"2022/09/19 03:15:11\",\"s\":\"1\"},...]") 或 hBarChart.setIOTData("[{\"x\":\"label1\",\"y\":\"12\",\"s\":\"1\"},...]")

Note: The data structure of the parameter 'data' must conform to the specifications outlined in "Bar Style >> Chart Type" above; otherwise, the data will be invalid.

### 24. updateIOTData（Getting IoT data and refreshing it）

```
/**
* @Description Manually refresh IoT data.
* @Return None*/
void updateIOTData();
```

Example:Form.hBarChart.updateIOTData()

## 6.21.8. Component macro properties

| Attribute | Type | Description | Example |
|---|---|---|---|
| backgroundColor | QColor | Background color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.backgroundColor=<br>HProperty.setQColor(colorMap) |
| barChartWidth | Int | Bar width | Form.hBarChart.barChartWidth=10 |
| dataLabelsColor | QColor | Bar chart data label text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.dataLabelsColor=H<br>Property.setQColor(colorMap) |
| dataLabelsFont | QFont | Bar chart data label font | var fontMap={};<br>fontMap["weight"]=12;<br>fontMap["pointSize"]=12;<br>fontMap["family"]="宋体";<br>fontMap["italic"]=true;<br>fontMap["underline"]=true;<br>Form.hBarChart.dataLabelsFont=H<br>Property.setQFont(fontMap) |
| dataLabelsVisible | bool | Bar chart data label visibility | Form.hBarChart.dataLabelsVisible=<br>true |

| | | | |
|---|---|---|---|
| displayDirection | int | Bar chart display direction, where 0 represents vertical and 1 represents horizontal | Form.hBarChart.displayDirection=0 |
| legendFont | QFont | Legend text font | var fontMap={};<br>fontMap["weight"]=12;<br>fontMap["pointSize"]=12;<br>fontMap["family"]="宋体";<br>fontMap["italic"]=true;<br>fontMap["underline"]=true;<br>Form.hBarChart.legendFont=HProperty.setQFont(fontMap) |
| legendPosition | int | Legend display position, where 0 represents topLeft, 1 represents topCenter, 2 represents topRight, 3 represents horizontalLeft, 4 represents horizontalRight, 5 represents bottomLeft, 6 represents bottomCenter, and 7 represents bottomRight | Form.hBarChart.legendPosition=1 |
| legendTextColor | QColor | Legend text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0; |

| | | | |
|---|---|---|---|
| | | | colorMap["blue"]=0;<br>Form.hBarChart.legendTextColor=H<br>Property.setQColor(colorMap) |
| legendVisible | bool | Legend visibility | Form.hBarChart.legendVisible=true |
| seriesSpacing | int | Bar chart series spacing | Form.hBarChart.seriesSpacing=2 |
| title | QString | Bar chart title | Form.hBarChart.title=”标题” |
| titleColor | QColor | Bar chart title text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.titleColor=HPropert<br>y.setQColor(colorMap) |
| titleFont | QFont | Bar chart title font | var fontMap={};<br>fontMap["weight"]=12;<br>fontMap["pointSize"]=12;<br>fontMap["family"]="宋体";<br>fontMap["italic"]=true;<br>fontMap["underline"]=true;<br>Form.hBarChart.titleFont=HProperty<br>.setQFont(fontMap) |
| titleVisible | bool | Bar chart title visibility | Form.hBarChart.titleVisible=true |
| xAxisColor | QColor | X-axis line color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.xAxisColor=HProp |

| | | | |
|---|---|---|---|
| | | | erty.setQColor(colorMap) |
| xAxisVisible | bool | X-axis line visibility | Form.hBarChart.xAxisVisible=true |
| xGridColor | QColor | X-axis grid line color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.xGridColor=HProperty.setQColor(colorMap) |
| xGridLineType | int | X-axis grid line style, where 0 represents dashed, 1 represents solid, and 2 represents dotted | Form.hBarChart.xGridLineType=1 |
| xGridVisible | bool | X-axis grid line visibility | Form.hBarChart.xGridVisible=true |
| xLabelRotation | int | Rotation angle of X-axis text labels | Form.hBarChart.xLabelRotation=10 |
| xTitle | QString | X-axis title | Form.hBarChart.xTitle=”X” |
| xTitleColor | QColor | X-axis title color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.xTitleColor=HProperty.setQColor(colorMap) |
| xTitleFont | QFont | X-axis title font | var fontMap={};<br>fontMap["weight"]=12;<br>fontMap["pointSize"]=12; |

| | | | |
|---|---|---|---|
| | | | fontMap["family"]="宋体";<br>fontMap["italic"]=true;<br>fontMap["underline"]=true;<br>Form.hBarChart.xTitleFont=HPrope<br>rty.setQFont(fontMap) |
| xTitleVisible | bool | X-axis title visibility | Form.hBarChart.xTitleVisible=true |
| xUpperEndi<br>ng | int | X-axis end point style,<br>where 0 represents<br>esNone, 1 represents<br>esFlatArrow, 2<br>represents<br>esSpikeArrow, 3<br>represents<br>esLineArrow, 4<br>represents esDisc, 5<br>represents esSquare, 6<br>represents esDiamond,<br>7 represents esBar, 8<br>represents esHalfBar,<br>and 9 represents<br>esSkewedBar | Form.hBarChart.xUpperEnding=1 |
| yAxisColor | QColor | Y-axis line color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.yAxisColor=HProp<br>erty.setQColor(colorMap) |
| yAxisMaxi | int | Y-axis maximum value | Form.hBarChart.yAxisMaximum=20 |

| | | | |
|---|---|---|---|
| mum | | | |
| yAxisMinimum | int | Y-axis minimum value | Form.hBarChart.yAxisMinimum=0 |
| yAxisStep | int | Y-axis tick step | Form.hBarChart.yAxisStep=5 |
| yAxisValRangeAutoAdjust | bool | Y-axis auto-fit to range | Form.hBarChart.yAxisValRangeAutoAdjust=true |
| yAxisVisible | bool | Y-axis line visibility | Form.hBarChart.yAxisVisible=true |
| yGridColor | QColor | Y-axis grid line color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.yGridColor=HProperty.setQColor(colorMap) |
| yGridLineType | int | Y-axis grid line style, where 0 represents dashed, 1 represents solid, and 2 represents dotted | Form.hBarChart.yGridLineType=1 |
| yGridVisible | bool | Y-axis grid line visibility | Form.hBarChart.yGridVisible=true |
| yTitle | QString | Y-axis title | Form.hBarChart.yTitle=”Y” |
| yTitleColor | QColor | Y-axis title text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.hBarChart.yTitleColor=HProp |

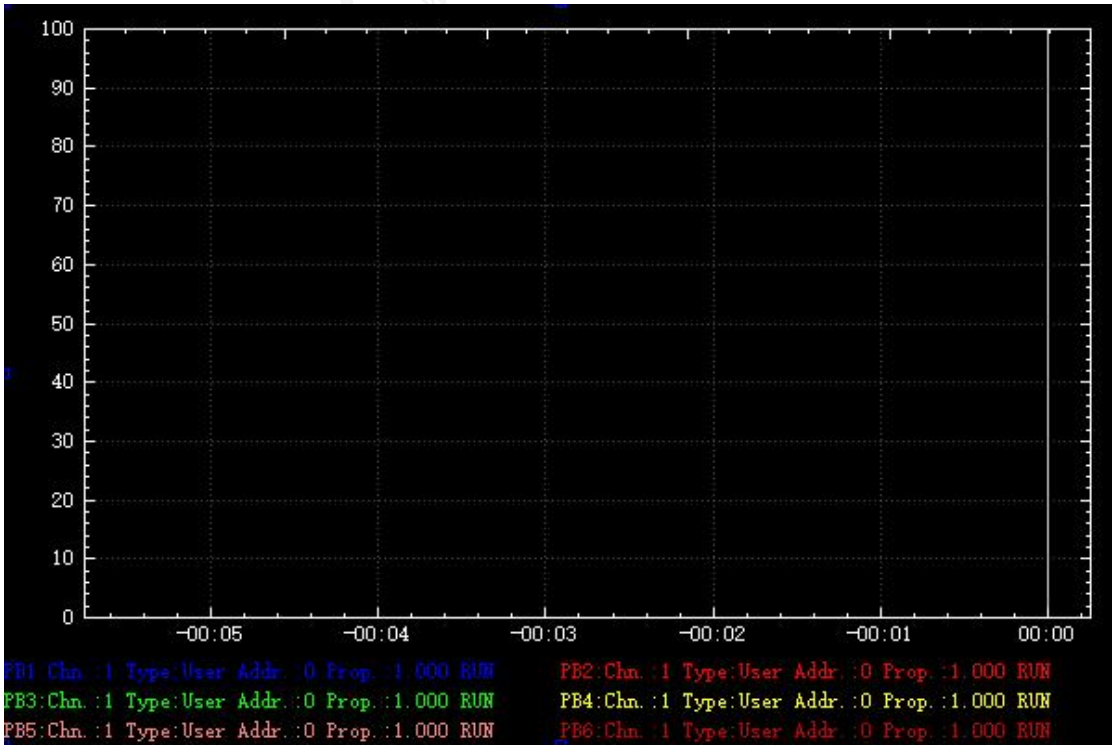
| | | | |
|---|---|---|---|
| | | | erty.setQColor(colorMap) |
| yTitleFont | QFont | Y-axis title font | var fontMap={};<br>fontMap["weight"]=12;<br>fontMap["pointSize"]=12;<br>fontMap["family"]="宋体";<br>fontMap["italic"]=true;<br>fontMap["underline"]=true;<br>Form.hBarChart.yTitleFont=HPrope<br>rty.setQFont(fontMap) |
| yTitleVisible | bool | Y-axis title visibility | Form.hBarChart.yTitleVisible=true |
| yUpperEndi<br>ng | int | Y-axis end point style, where 0 represents esNone, 1 represents esFlatArrow, 2 represents esSpikeArrow, 3 represents esLineArrow, 4 represents esDisc, 5 represents esSquare, 6 represents esDiamond, 7 represents esBar, 8 represents esHalfBar, and 9 represents esSkewedBar | Form.hBarChart.yUpperEnding=1 |
| zoomEnabl<br>ed | bool | Enable zoom for bar chart | Form.hBarChart.zoomEnabled=true |

# 6.22. Variable Oscilloscope

## 6.22.1. Function Introduction

The variable oscilloscope continuously samples variables, converting them into visual waveform graphs to assist engineers and technicians in analyzing problems.

## 6.22.2. Component Structure




**1. Numerical Axis**

The numerical axis displays the variable values at corresponding positions.

**2. Time Axis**

The time axis represents the variation of variables over time on the oscilloscope's horizontal axis.

**3. Probe Information**

Displays probe information, supporting up to six probes.

Number:PB1:

Channel:Chn: 1

Address Type: User

Address:Addr: 0

Scale:Prop: 1.000

Status:RUN Probe

## 6.22.3. Property Editor

**1. Refresh Time**

Set the refresh period for drawing, in milliseconds (ms). A smaller value increases refresh speed but may consume more CPU resources.

**2. Time Range**

Set the time length for the time axis, in seconds (s).

**3. Time Format**

1) HHMMSS

2) MMSS

3) SS

**4. Time Axis Visibility**

Set whether the time axis is visible.

**5. Vertical Ruler Minimum Value**

Set the minimum value for the numerical axis. Adjusting the minimum and maximum values of the axis ensures complete waveform display when the waveform is not within

the viewport range.

**6. Vertical Ruler Maximum Value**

Set the maximum value for the numerical axis. Adjusting the minimum and maximum values of the axis ensures complete waveform display when the waveform is not within the viewport range.

**7. Vertical Ruler Visibility**

Set whether the vertical axis is visible.

**8. Background Color**

Set the background color.

**9. Zero Line Color**

Set the color of the zero scale line on the numerical axis.

**10. Grid Line Color**

Set the color of the grid lines.

**11. Grid Display**

Set whether the grid lines are displayed.

**12. PBx Display**

Set whether probe x is displayed.

**13. PBx Channel**

Set the channel used by probe x.

**14. =PBx Data Type**

Set the data type of probe x.

**15. PBx Address**

Set the address of probe x.

**16. PBx Line Width**

Set the waveform line width of probe x.

**17. PBx Color**

Set the color of probe x.

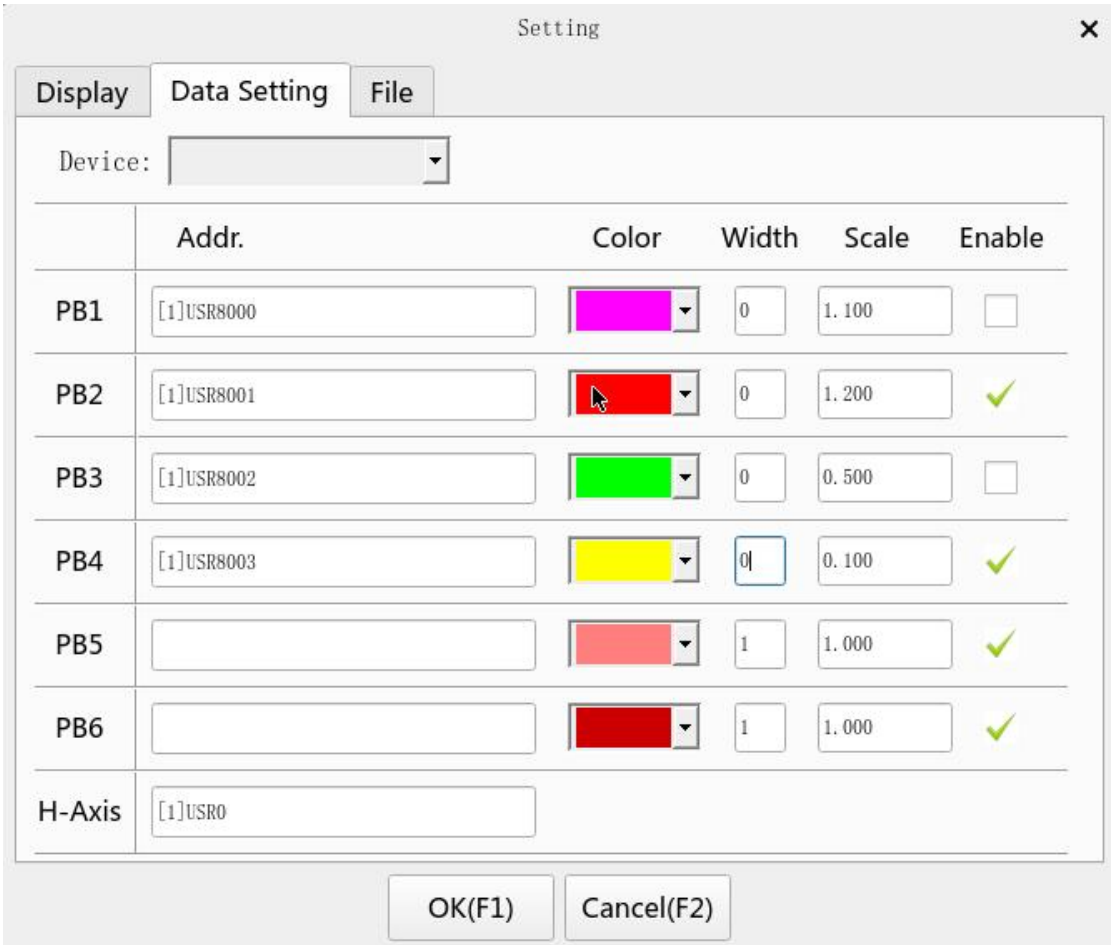
**18. Device**

Set the device that the component connects to when using remote visualization.

## 6.22.4. Parameter Settings Popup

During runtime, the macro paraDataSetDialog can be used to bring up the parameter settings popup. The popup has three pages: "Display", "Numeric Settings", and "File Settings".

[Display]




**1. Refresh Period**

Set the refresh period for drawing, in milliseconds (ms). A smaller value increases refresh speed but may consume more CPU resources.

**2. Horizontal Axis**

1) Horizontal Axis Style

Invalid property.

2) Display Duration

Set the time length for the time axis, in seconds (s).

3) Style

a. HHMMSS

b. MMSS

c. SS

4) Visibility

Set whether the horizontal axis is visible.

**3. Vertical Axis**

1) Minimum Value

Set the minimum value for the numerical axis.

2) Maximum Value

Set the maximum value for the numerical axis.

3) Visibility

Set whether the vertical axis is visible.

**4. Zero Line**

Set the color of the zero line on the vertical axis.

**5. Background**

Set the background color.

**6. Grid**

Set the grid color.

**7. Probe Fit (ZoomAuto)**

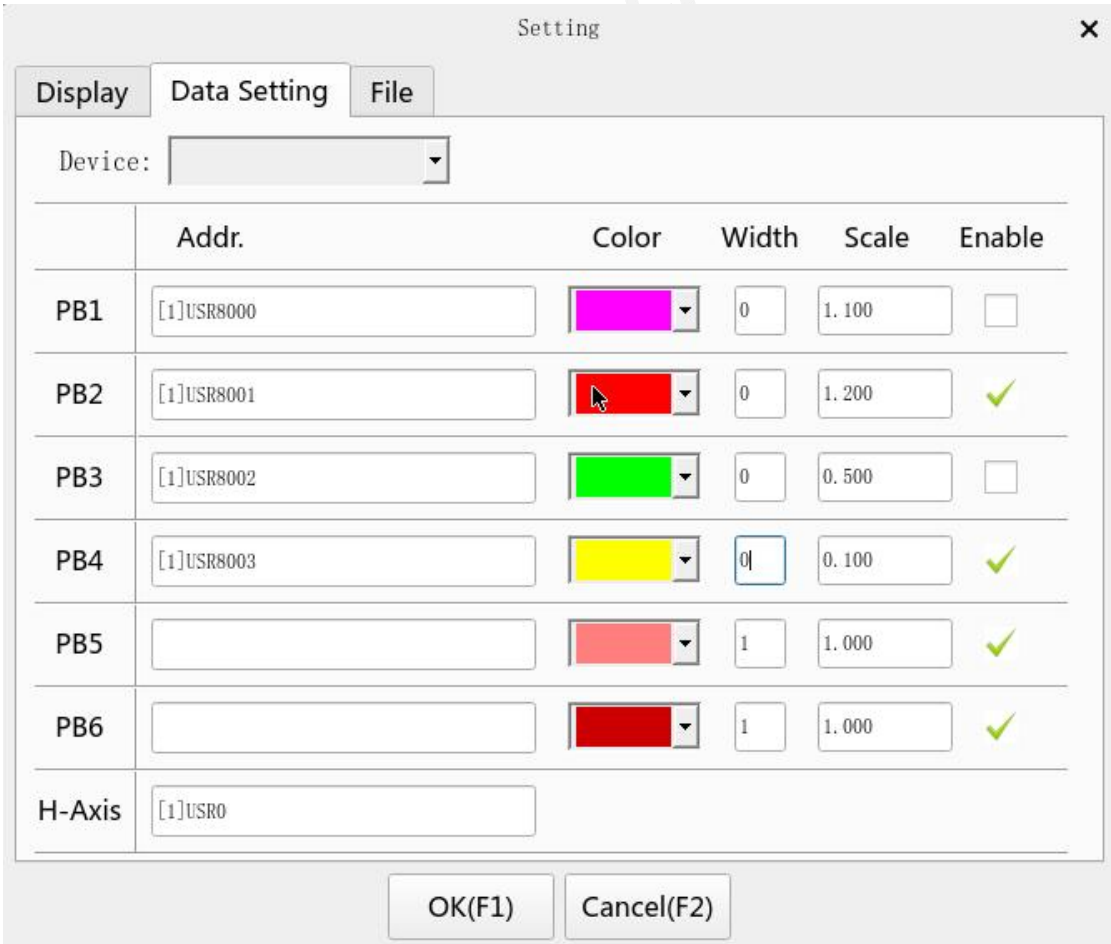
Set the axis to fit the probe, ensuring the waveform is fully displayed within the viewport.

Takes effect after calling setPBZoomAuto.

### 8. Decimal Places

Set the number of decimal places for the vertical axis.

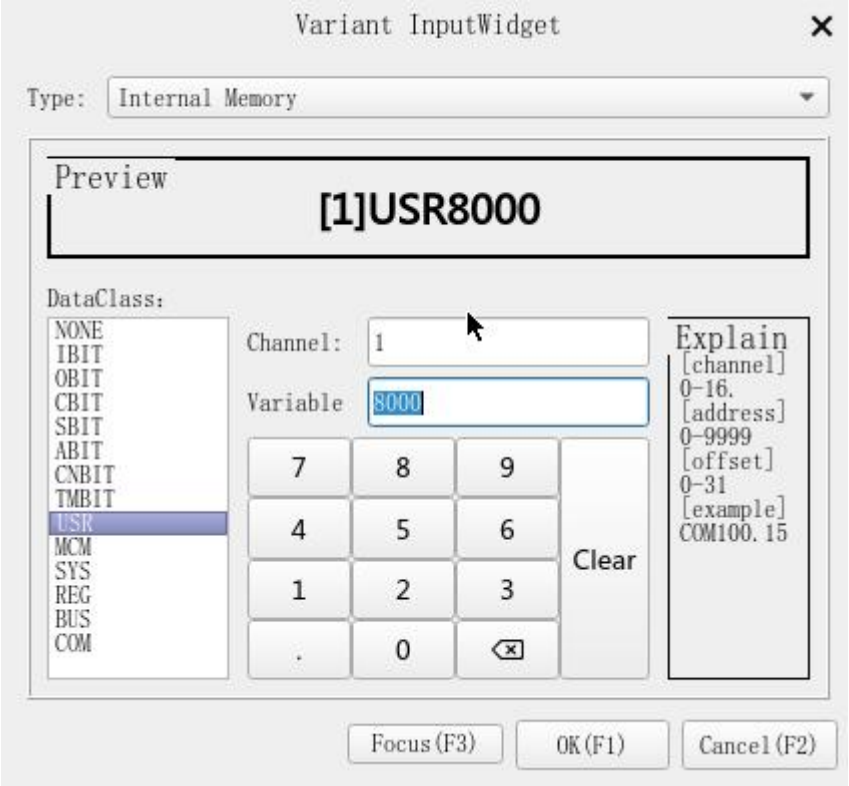
[Numeric Settings]




### 1. Device

Set the device that the component connects to when using remote visualization.

### 2. Address




When clicking on the address bar, use the variable input popup to edit the variable address. See details in "Variable Input Popup."

### 3. Color

Set the color of the waveform corresponding to the probe

### 4. Line Width

Set the line width of the waveform corresponding to the probe.

### 5. Scale

Set the scale of the waveform corresponding to the probe.

### 6. Enable

Set whether the probe is enabled. Waveform plotting requires the probe to be enabled.

### 7. H-Axis

Invalid parameter.

[File Settings]

Planned functionality to save waveform data to files and load waveform data from files. Currently not implemented.

## 6.22.5. Script Macro

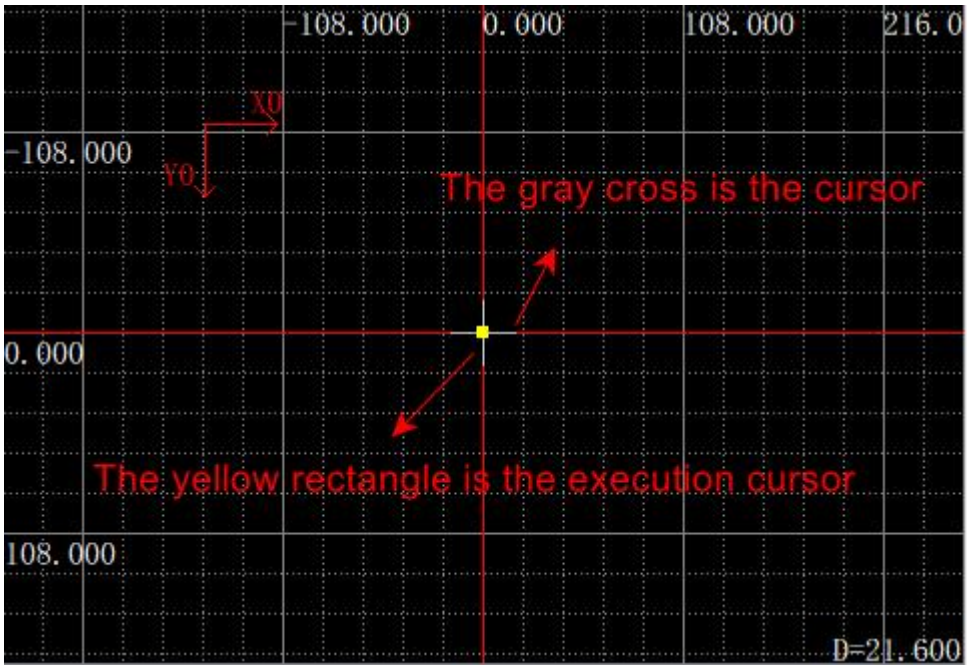
See details in "HCurvsGraphics_A4_Designer_CN.xml".

# 6.23. Plotting

## 6.23.1. Function Introduction

The plotting component has two main functions: static plotting and dynamic plotting. Static plotting is used to preview program trajectories, while dynamic plotting draws program trajectories during actual runtime.

## 6.23.2. Component Structure



**1. Axis Description**

Axial direction typically refers to the coordinate axes of a machine tool, indicating the current axis orientation that is set.

**2. Zero scale line**

Zero scale line refers to the reference line on a ruler or scale, typically used to establish the starting point or reference position for measurements. It is usually depicted as a distinct line or mark, serving to establish the initial position.

**3. Scales**

Scales are the evenly spaced lines used to mark positions in a graph. In plotting, scales are divided proportionally to help determine positions accurately.

**4. Grid lines**

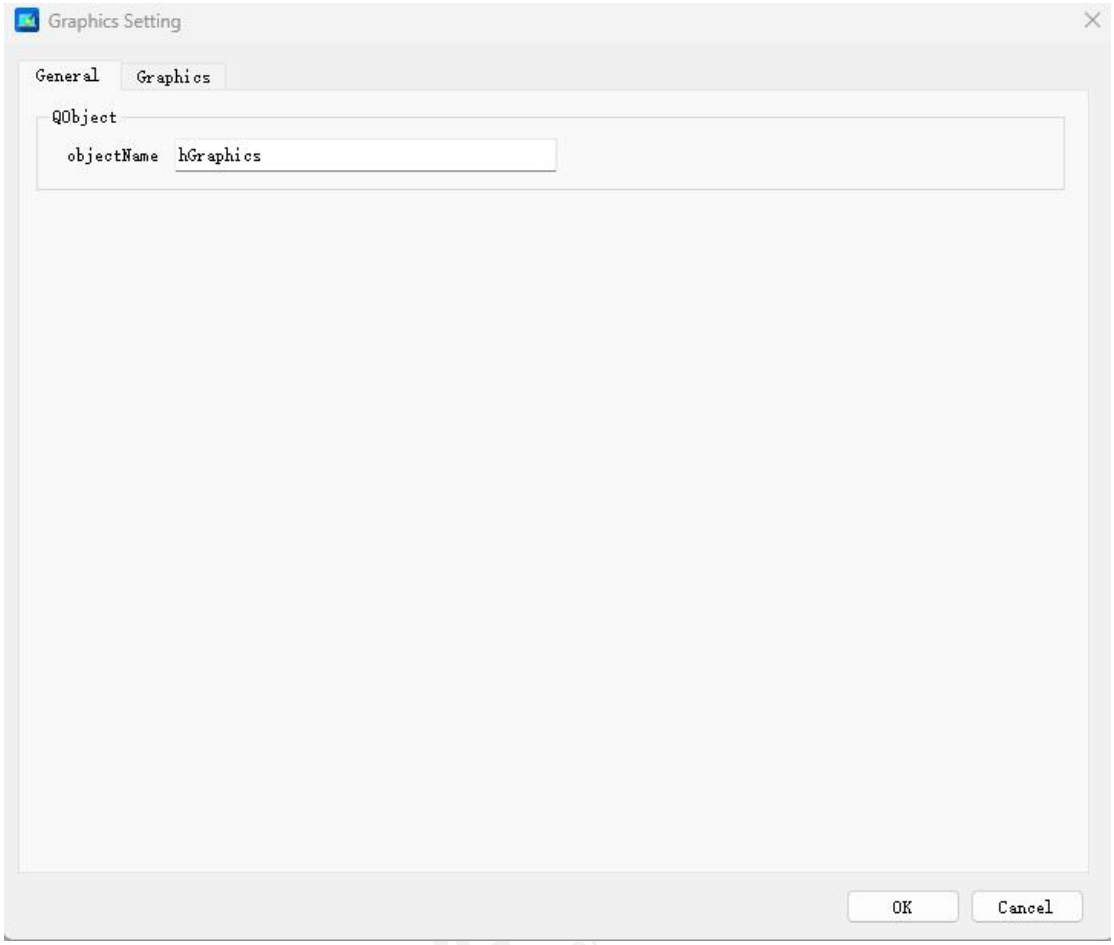
Grid lines are reference lines used in plotting, appearing as evenly spaced parallel lines.

**5. Scale Value**

The actual step size between grid lines.

## 6.23.3. The actual step size between grid lines.

In the HMI software editing interface, double-clicking a component with the left mouse button opens the property setting popup. The property setting popup typically includes two tabs: "Basic Properties" and "Plotting".




[General Properties]

**1. Object Name**

Each object can be identified and accessed by setting its object name (Object Name). An object name is a string used to reference and manipulate objects in code.

**2. Geometric Dimensions**

This attribute controls the position and size of the component within the parent window. Refer to "Geometric Dimensions" for parameter details.

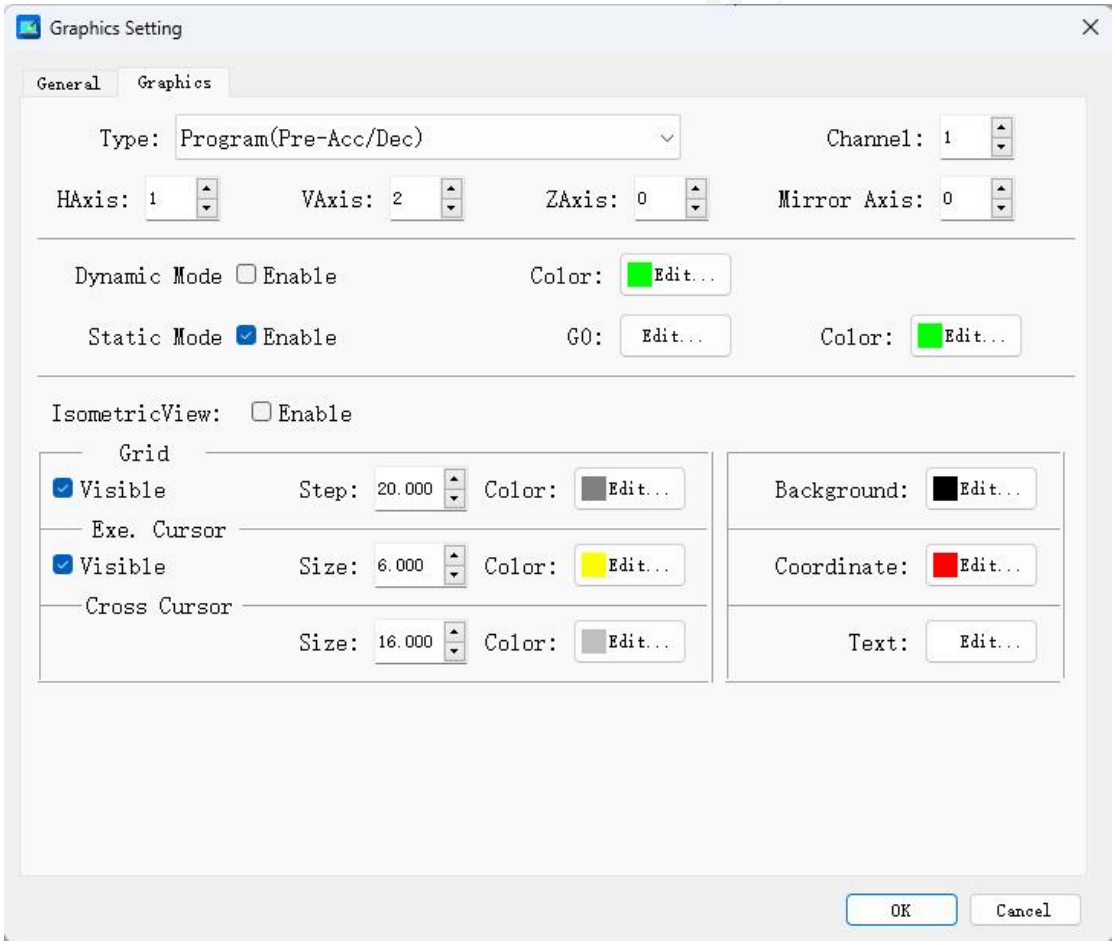
**3. Text**

Sets the font of the component's text. Refer to "Font" for parameter details.

### 4. Border

Sets the border of the component. Refer to "Frame" for detailed parameter explanations.

[Plotting]




### 1. Data Type

Set the data source for plotting. It can be set to the following options:

1) Program Coordinates (Before Acceleration/Deceleration)
2) Program Coordinates (After Acceleration/Deceleration)
3) Machine Coordinates
4) Relative Coordinates
5) Feedback Coordinates

**2. Channel**

Specify the channel for the data source of plotting.

**3. Axial**

1) Horizontal axis

Specify the actual machine axis corresponding to the horizontal axis of the plotting.

2) Vertical axis

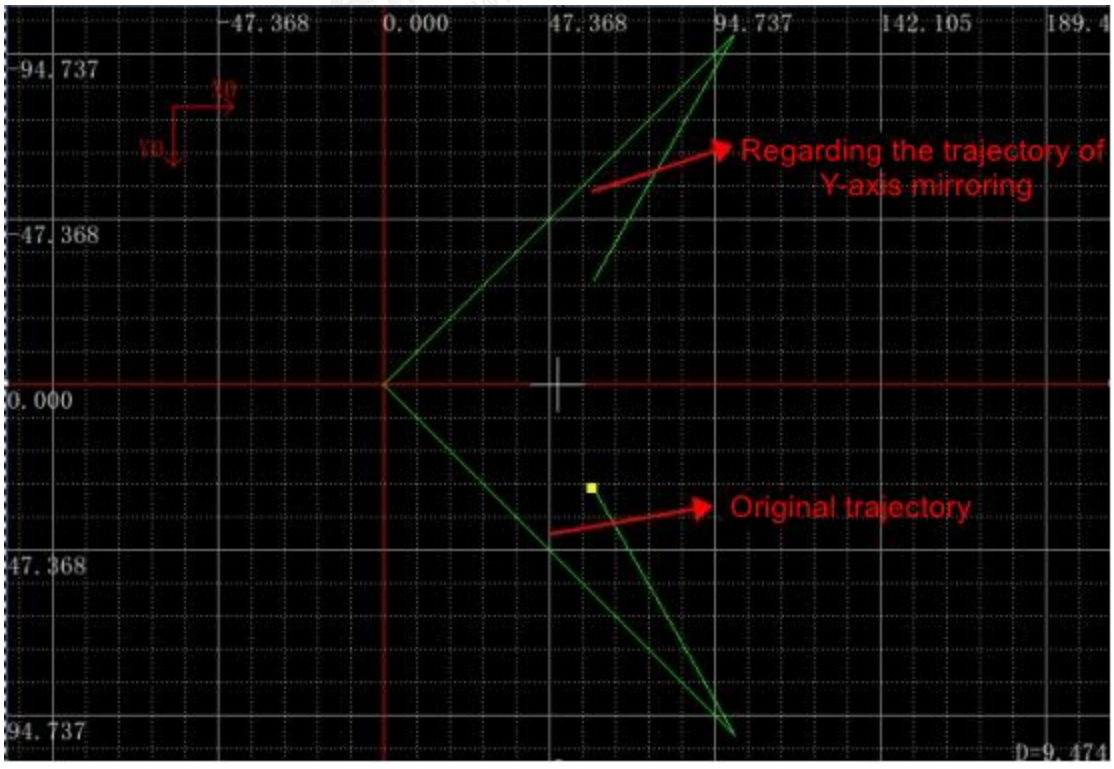
Specify the actual machine axis corresponding to the vertical axis of the plotting.

3) Vvertical axis

Specify the actual machine axis corresponding to the vertical axis of the plotting.

4) Mirror Axis

Specify the mirroring axis for plotting. When a mirroring axis is specified, the plot will be mirrored accordingly during drawing.



## 1. Drawing

1) Dynamic Tracing

To plot the actual trajectory of a program during runtime, you can enable dynamic plotting and set the color for the dynamic plot trajectory.

2) Static Tracing

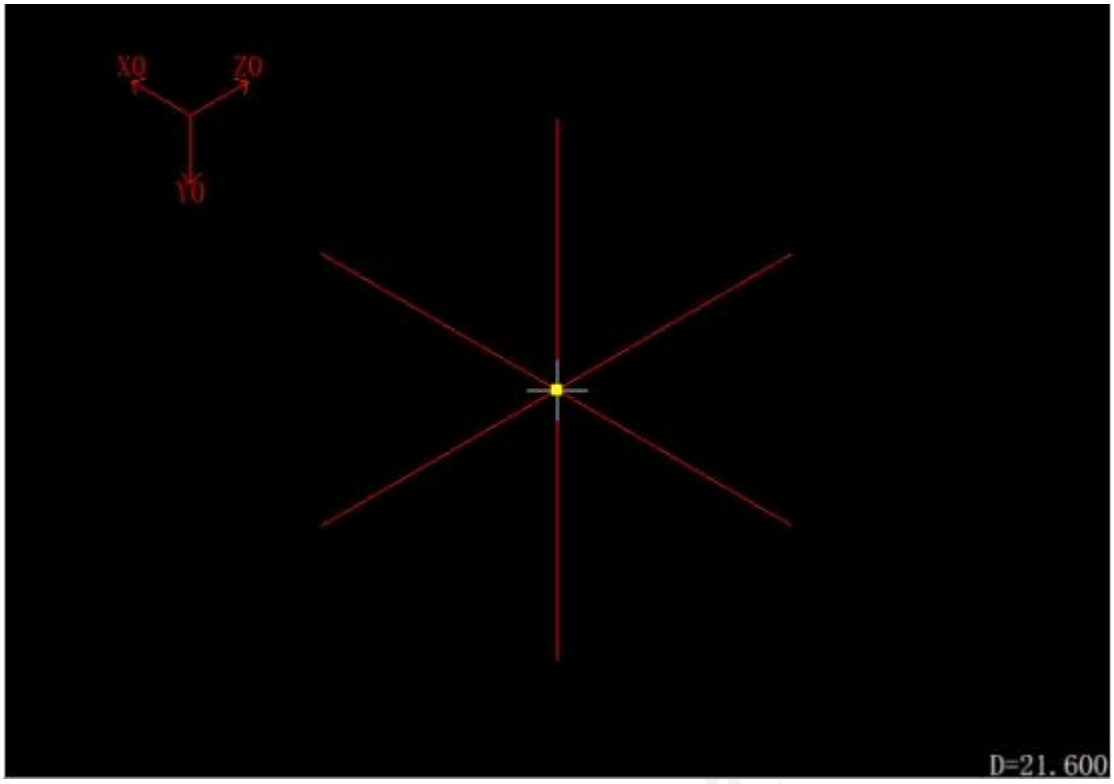
Preview the program's running trajectory. Static plotting allows you to set colors for G0 codes and other G codes to differentiate between rapid positioning and machining trajectories.

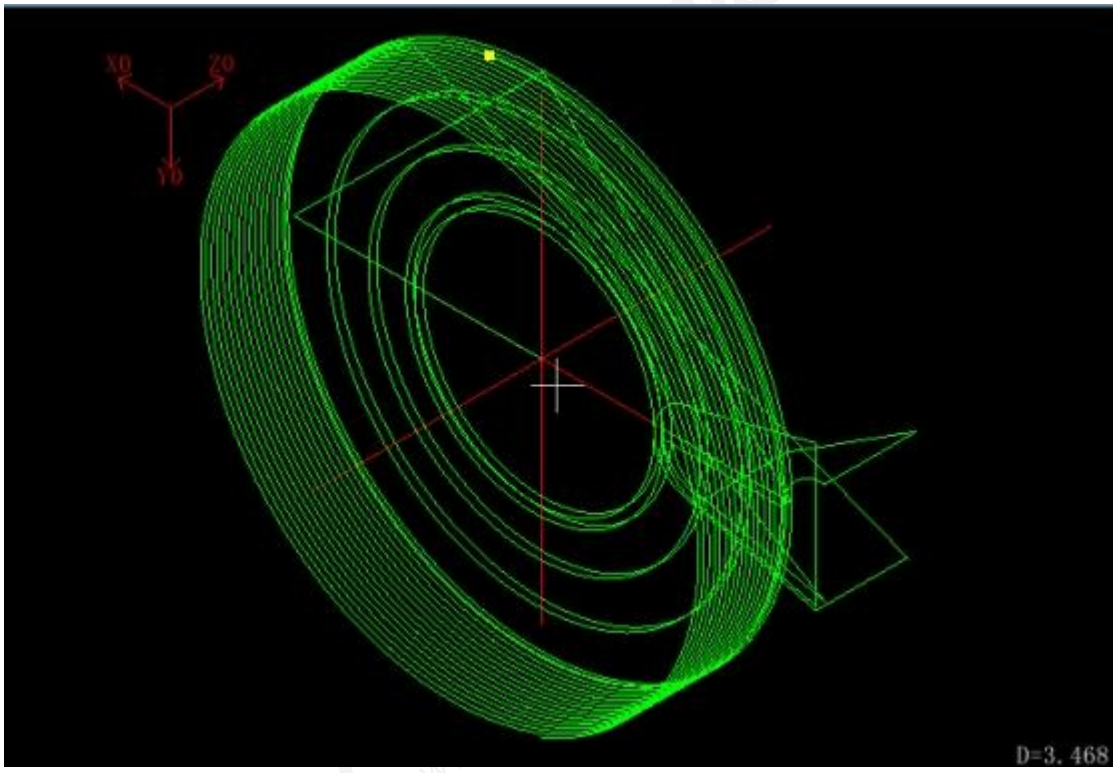
## 2. Viewport

3) Isometric Display

Isometric projection is a method used to represent three-dimensional objects graphically, allowing them to be depicted on a two-dimensional plane. In isometric projection, the three coordinate axes of the object (typically x, y, and z axes) intersect at equal angles on the drawing, providing a more intuitive and realistic representation of the object on paper.

Isometric View (Note: Isometric projection requires setting the horizontal axis, vertical axis, and vertical axis before it can take effect.)




4) Background

Set the viewport background color.

5) Zero line

Set the zero line color.

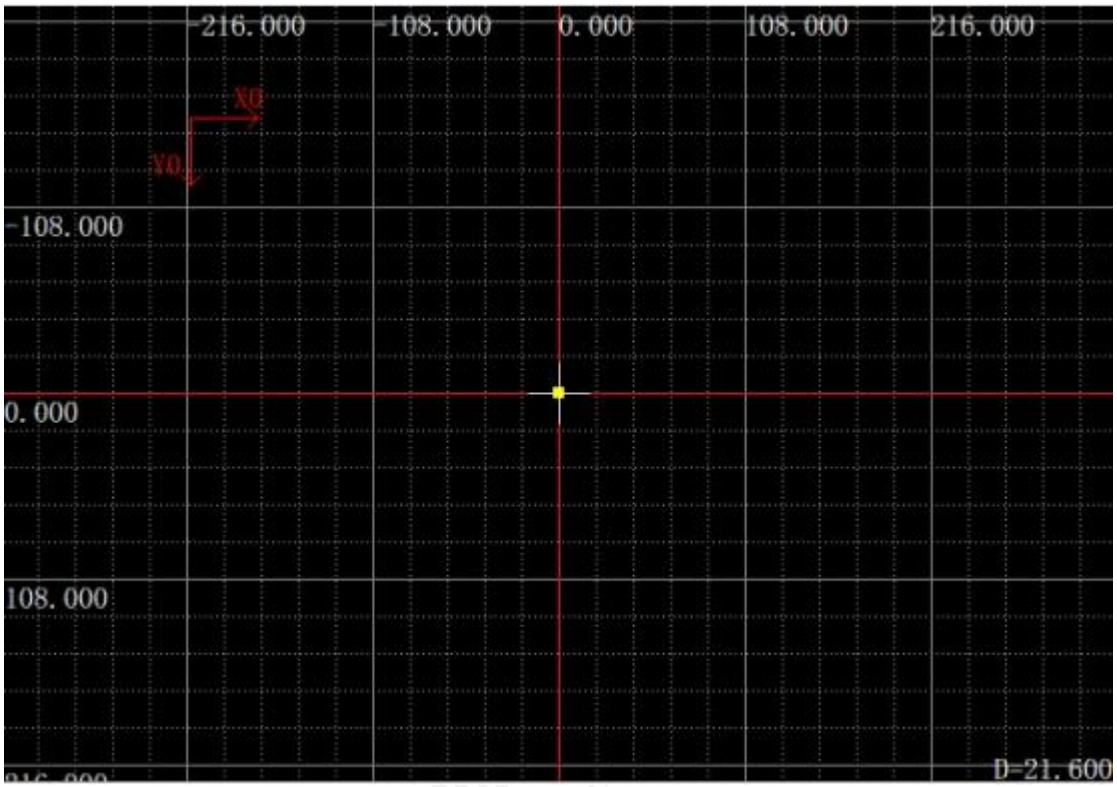
6) Text

Set the text color.

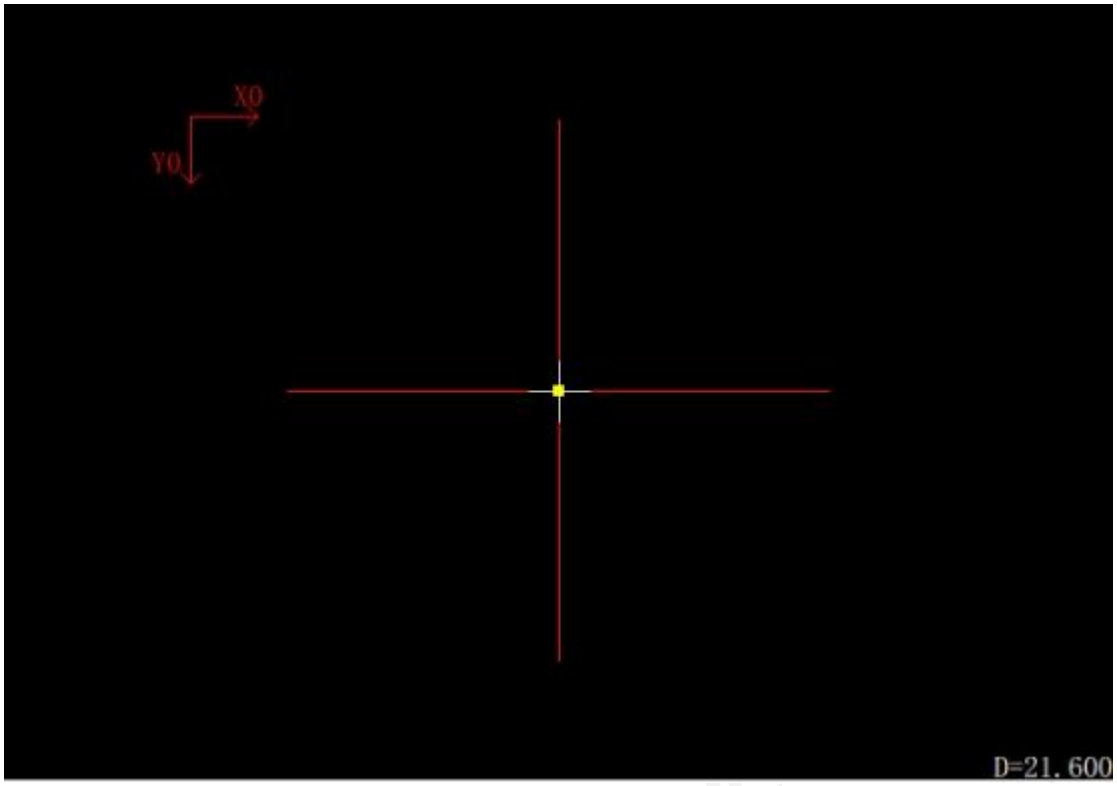
**3. Grid**

7) Display

Display the grid style (note that the grid will automatically hide after setting the vertical axis).




Style when grid is not displayed:




8) Step Size

Set the grid step size. The actual grid step size scales based on the zoom ratio. Therefore, the actual grid step size is represented by the "scale value".


9) Color

Set the grid line color.

4. **Execution Cursor**

During dynamic plotting, the execution cursor indicates the current position being drawn.

10) Display

Set whether the execution cursor is displayed.

11) Size

Set the size of the execution cursor.

12) Color

Set the color of the execution cursor.

**5. Static Cursor**

During static plotting, the static cursor indicates the current position being drawn.

13) Display

Set whether the static cursor is displayed

14) Size

Set the size of the static cursor.

15) Color

Set the color of the static cursor

**6. Cursor**

16) Size

Set the cursor size.

17) Color

Set the cursor color.

## 6.23.4. Property Editor

The property editor provides a convenient way to set properties without the need to double-click and open the property setting popup. Most of the property settings in the property editor are equivalent to those in the property setting popup. Here, we will only introduce functions not covered in the property setting popup.

**1. Refresh Rate**

Set the refresh rate for drawing. A smaller value results in faster refresh rates but

increases CPU resource consumption.

### 2. Cursor Movement Amount

The distance moved each time when using the "Up", "Down", "Left", and "Right" macros to move the cursor (in pixels).

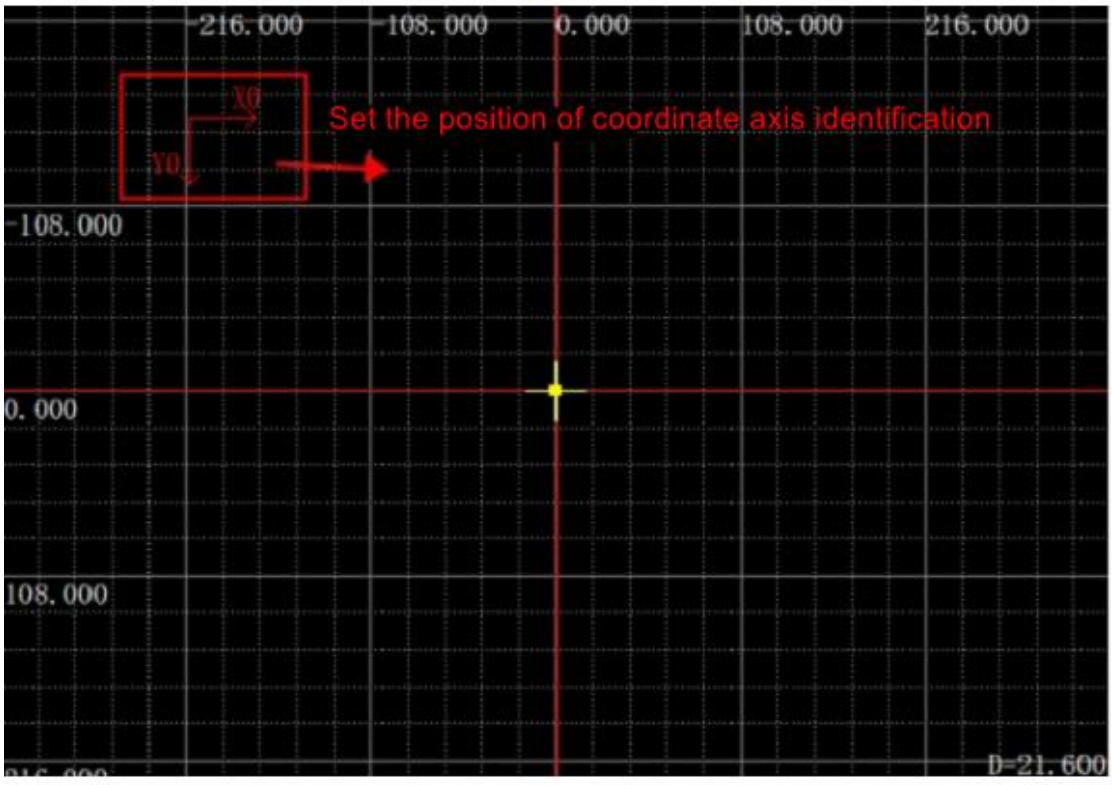
### 3. Center Movement Amount

The distance moved each time when using the "Up", "Down", "Left", and "Right" macros to move the viewport (in pixels).

### 4. Zoom Increment

The scaling ratio each time the viewport is zoomed in or out using the "Zoom In" and "Zoom Out" macros.

### 5. Origin Position



### 6. Cursor Shape

The cursor can be set to Rect (rectangle) or Circle (circle) for execution.

### 7. Cursor Color

That would be "Cursor Color"

### 8. Dxf color

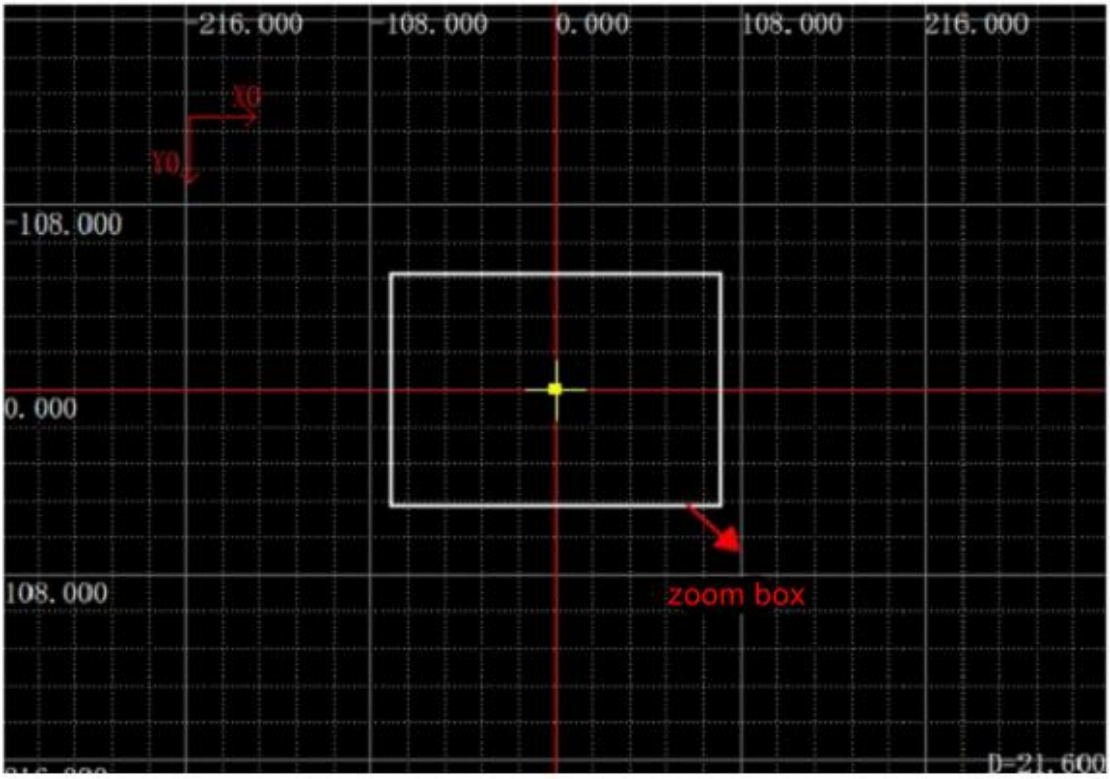
Invalid property

### 9. Plot Color

Invalid property

### 10. Zoom Box Color

The zoom box is used to zoom into a selected area. You can set the color of the zoom box.



**11. Zoom Box Line Width**

Set the line width of the zoom box.

**12. Zoom Box Movement Increment**

When moving the zoom box using "Up", "Down", "Left", or "Right", the distance moved each time is equal to the viewport width multiplied by the zoom box movement increment.

**13. Zoom Box Zoom Increment**

The scaling ratio for the zoom box when zooming in or out using "Zoom In" and "Zoom Out".

**14. Zoom Box Line Type**

The line type of the zoom box can be set to SolidLine (solid line) or DashLine (dashed line).

**15. Device**

Set the device that the component connects to when using remote visualization.

### 6.23.5. Script Macros

Details can be found in "HGraphics_Designer_CN.xml".

## 6.24. Line Chart

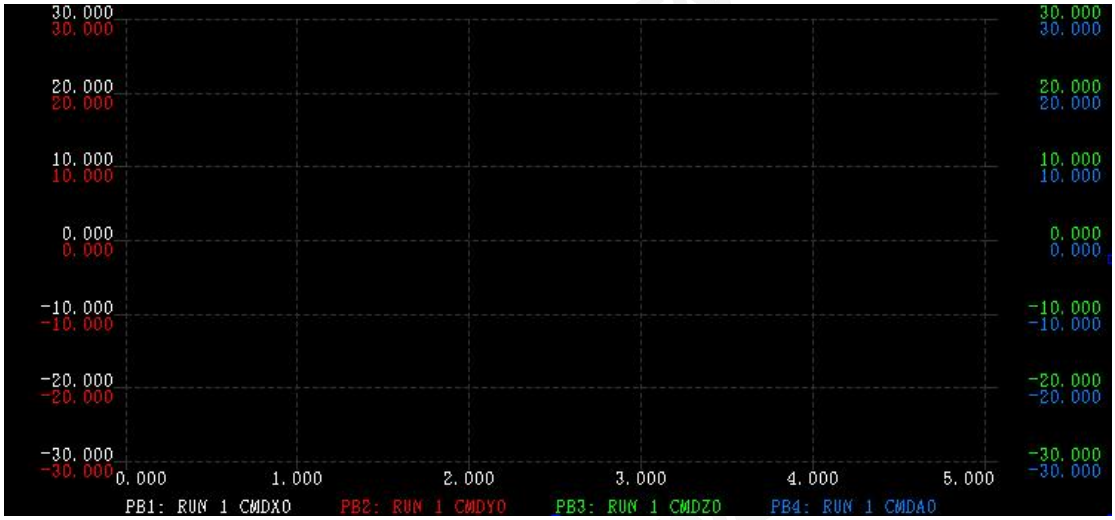
## 6.25. Oscilloscope

### 6.25.1. Function Introduction

The oscilloscope continuously samples axial data during program execution, converting

digital signals into visual waveform graphs to assist engineers and technicians in analyzing issues.

## 6.25.2. Component Structure




**1. Vertical Axis**

The vertical axis represents the numerical axis, displaying data values in different colors corresponding to each data point

**2. Horizontal Axis**

The horizontal axis represents the time axis, used to depict changes in data over time.

**3. Grid**

Grids are reference lines used in plotting, appearing as evenly spaced parallel lines.

**4. Probe Information**

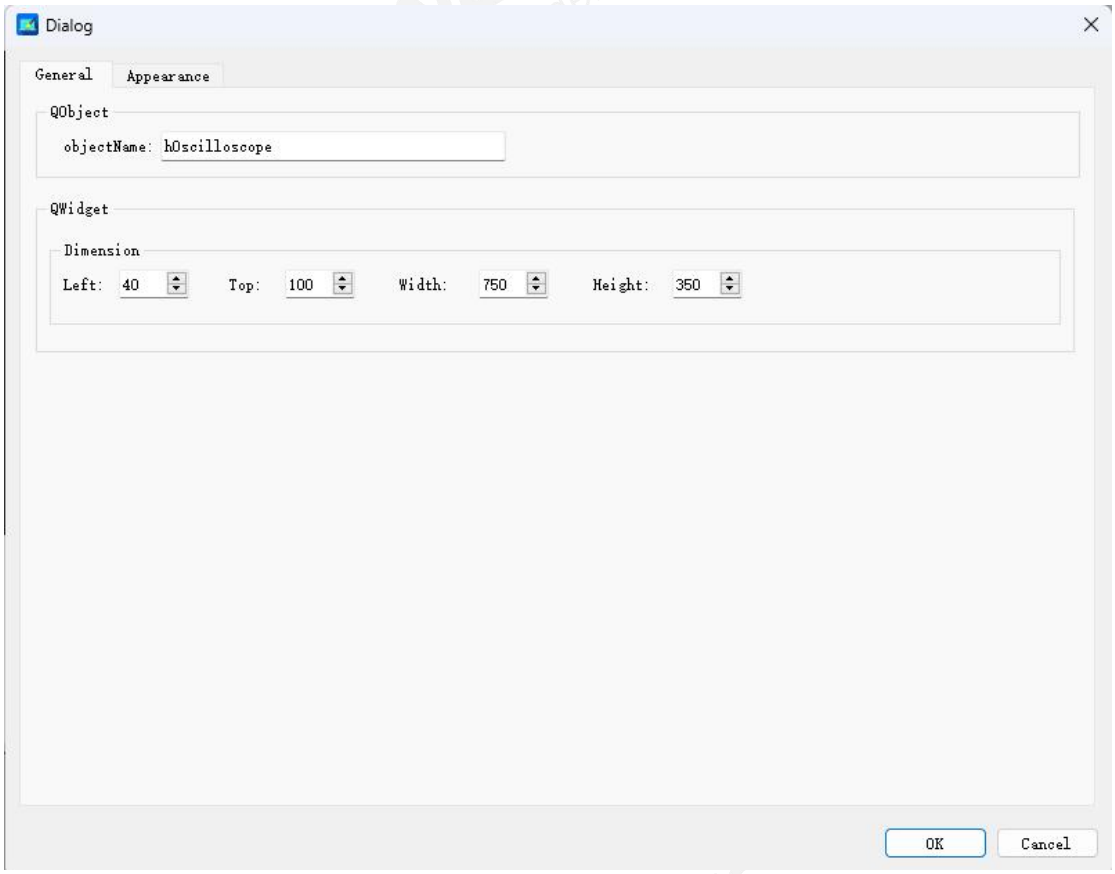
Display probe information, currently supporting up to four probes.

| PB1： | RUN | 1 | CMD | X0 |
|---|---|---|---|---|
| Probe number | Operation Status | Channel | Type | Axis |

## 6.25.3. Property Editing Popup

In the HMI software editing interface, double-clicking a component with the left mouse button opens the property setting window. The property setting window typically includes two tabs: "General" and "Appearance."

[General]




### 1. Object Name

Each object can be identified and accessed using an object name. An object name is a string used to reference and manipulate objects in code.

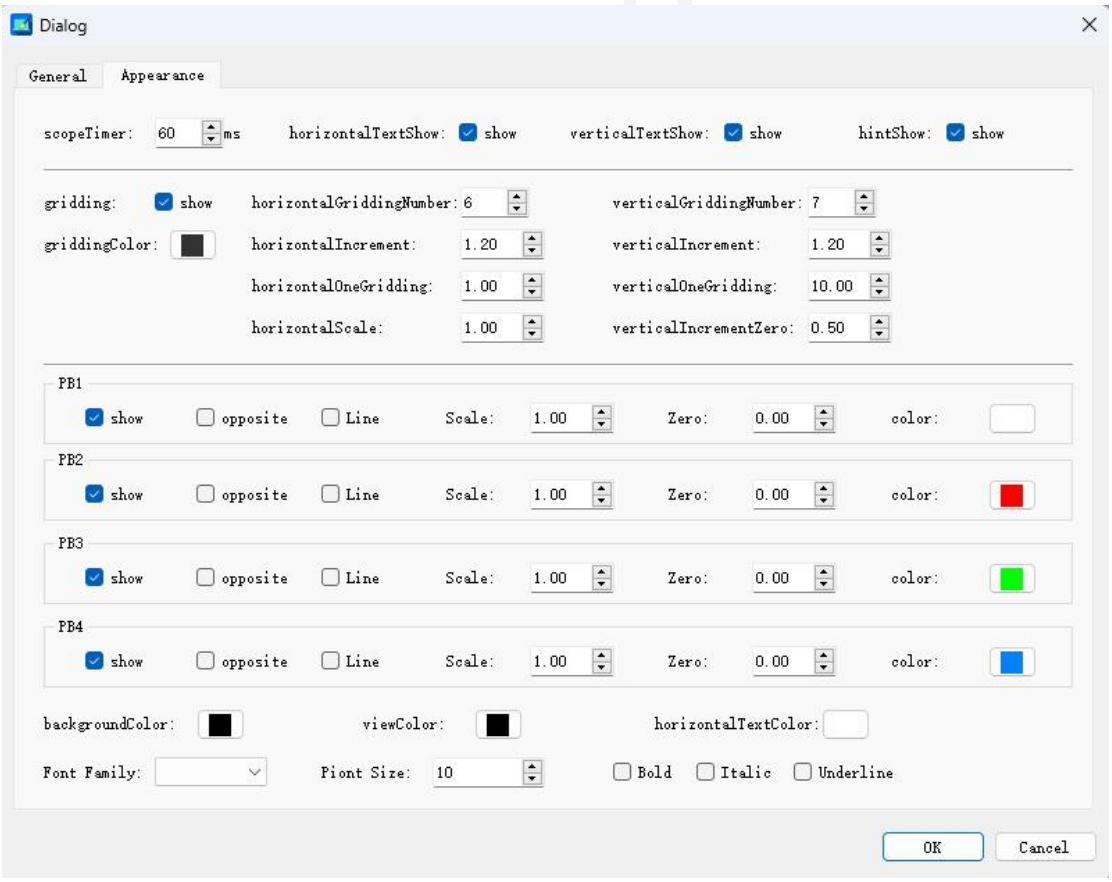
### 2. Geometry Dimensions

This property controls the position and size of the component within its parent window. Refer to "Geometry Dimensions" for detailed parameter explanations.

### 3. Text

Set the text font for the component. Refer to "Font" for detailed parameter explanations.

[Appearance]

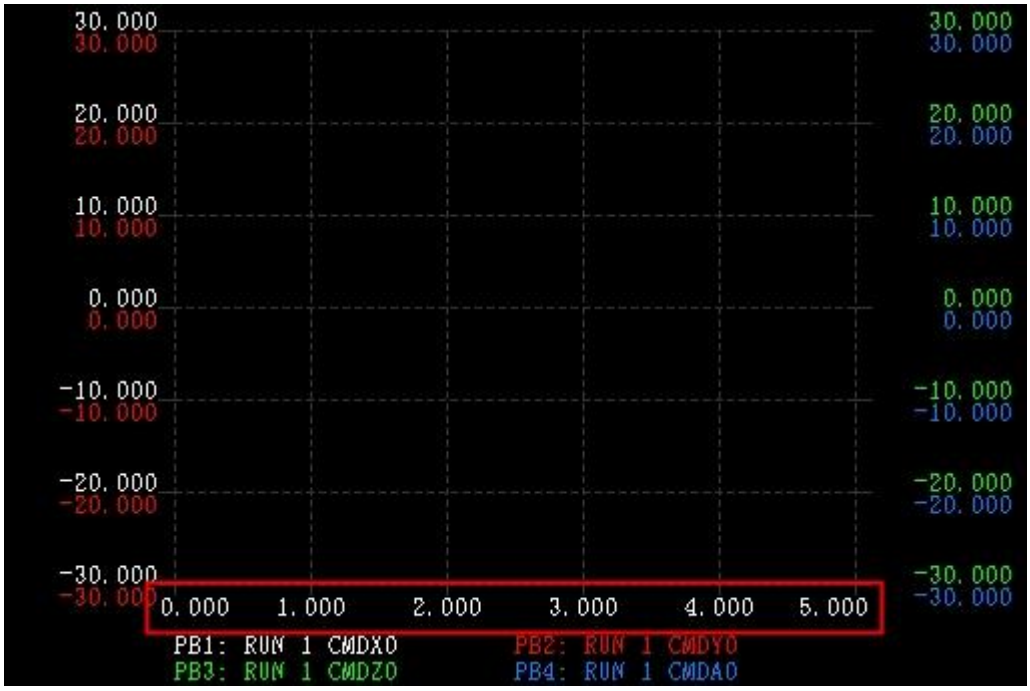



### 1. Refresh Period

Set the refresh period for drawing, in milliseconds. A smaller value results in faster refresh rates but increases CPU resource consumption.

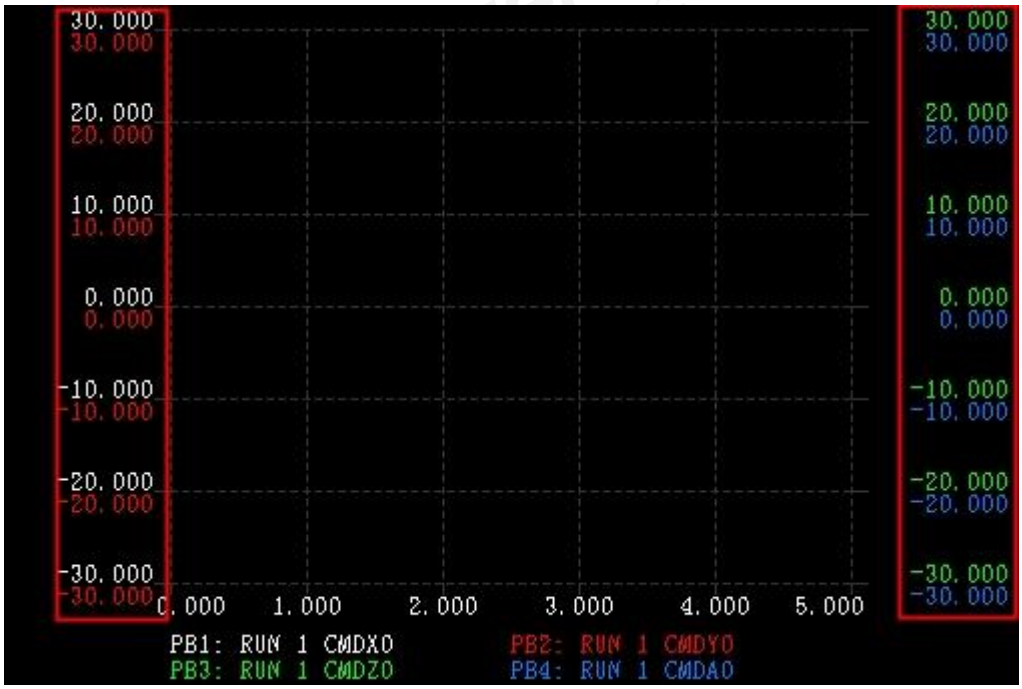
### 2. Horizontal Text

Horizontal Text Display



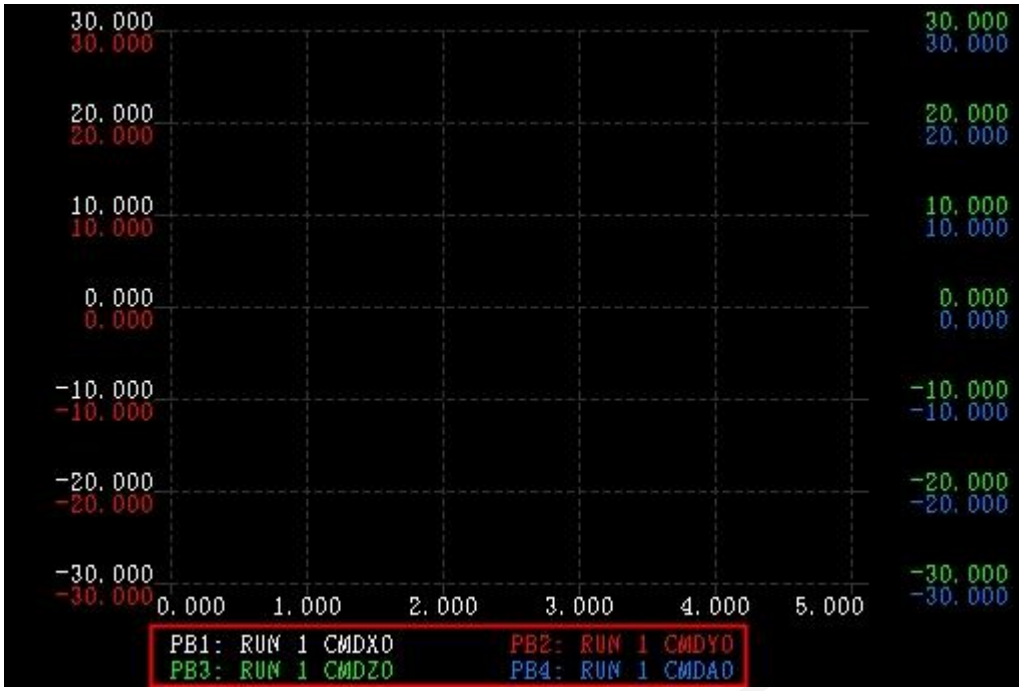

### 3. Vertical Text Display

Set whether the vertical axis text is displayed.




### 4. Probe Identifier Display

Set whether the probe identifier is displayed.




### 5. Grid Display

Set whether the grid is displayed.

### 6. Grid Color

Set the grid color

### 7. Horizontal Grid Count

Set the number of horizontal grids.

### 8. Horizontal Zoom Increment

Set the horizontal zoom increment, where the scaling ratio of the horizontal axis changes by multiplying the current scaling ratio with the horizontal zoom increment when using the "Zoom In" and "Zoom Out" macro methods on the horizontal axis.

### 9. Horizontal Step Size

Set the horizontal grid step size.

**10. Horizontal scaling ratio**

Set the horizontal scaling ratio.

**11. Vertical Grid Count**

Set the number of vertical grids.

**12. Vertical zoom increment**

Set the vertical zoom increment, where the scaling ratio of the vertical axis changes by multiplying the current scaling ratio with the vertical zoom increment when using the "Zoom In" and "Zoom Out" macro methods on the vertical axis.

**13. Vertical Step Size**

Set the vertical step size.

**14. Vertical movement increment**

Set the vertical movement increment, which determines the distance (in pixels) moved each time when using the "Up" and "Down" macro methods to move the vertical axis.

**15. PBx Display**

Enable probe data display to plot waveforms.

**16. PBx Invert**

Set whether probe data is inverted.

**17. PBx Line**

Set whether probe data is connected with straight lines. When this option is not selected, probe data will be plotted as a scatter plot.

### 18. PBx Scale

Set the scaling ratio for probe data.

### 19. PBx zero line

Set the zero line position for probe data.

### 20. PBx color

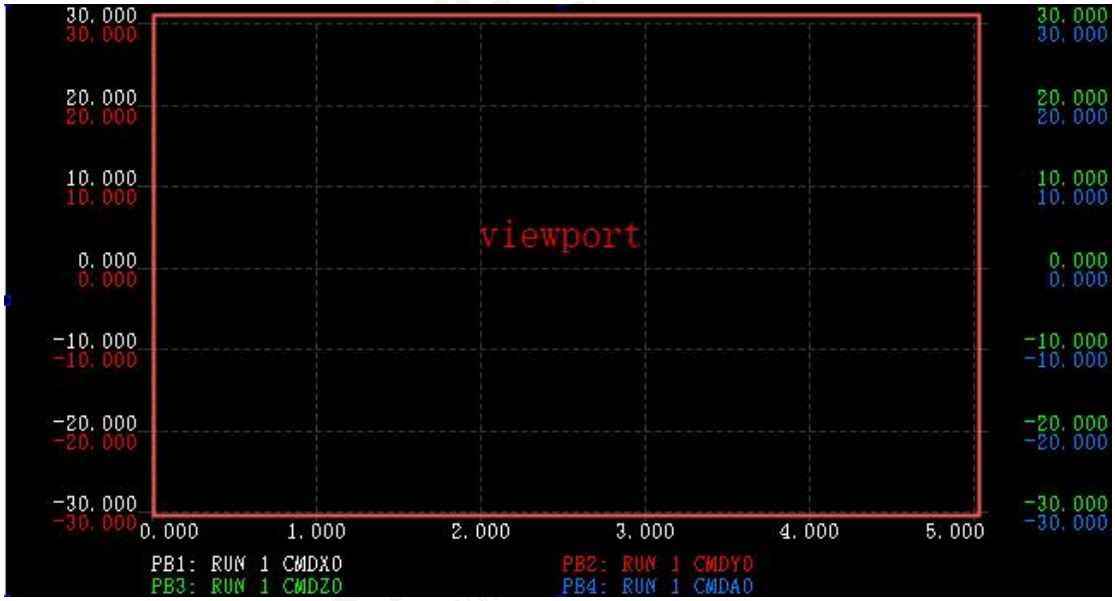
Set the color of the probe waveform.

### 21. Background Color

Set the background color.

### 22. Viewport Color

Set the color of the viewport.




### 23. Horizontal text color

Set the color of horizontal text.

**24. Font**

Set the font for all text. Refer to "Font" for detailed parameter information.

## 6.25.4. Script Macros

"Details can be found in 'HOscilloscope_Designer_CN.xml'."

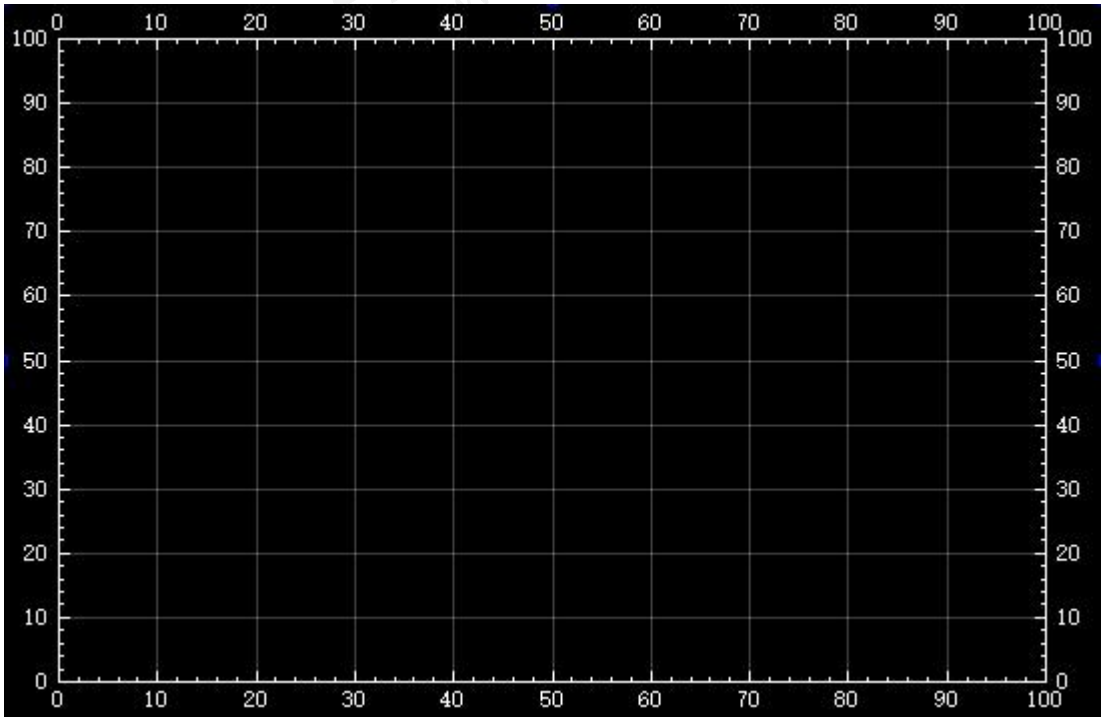
## 6.26. Custom Drawing

## 6.27. Feature Introduction

## 6.27.1. Feature Introduction

Custom drawing components allow users to create custom graphics such as lines, arcs, and rectangles.

## 6.27.2. Component Structure




**1. Horizontal axis**

In a coordinate plane, the horizontal axis (x-axis) typically intersects the vertical axis (y-axis) at a right angle, together forming a two-dimensional coordinate system used to describe the positions of points in a plane.

**2. Vertical axis**

In a coordinate plane, the vertical axis typically intersects the horizontal axis (x-axis) at a right angle, together forming a two-dimensional coordinate system used to describe the positions of points in a plane.

**3. Grid**

A grid is a set of evenly spaced parallel lines used as reference lines in graphical representations.

## 6.27.3. Property Editor

**1. Horizontal Step Size**

Horizontal step size refers to the distance between adjacent ticks on the horizontal axis within a coordinate system or chart.

**2. Vertical Step Size**

Vertical step size refers to the distance between adjacent ticks on the vertical axis within a coordinate system or chart.

**3. Horizontal Ruler Minimum Value**

The minimum value of the horizontal ruler refers to the smallest scale value displayed on the horizontal ruler.

**4. Horizontal Ruler Maximum Value**

The maximum value of the horizontal ruler refers to the highest scale value displayed on

the horizontal ruler.

**5. Vertical Ruler Minimum Value**

The minimum value of the vertical ruler refers to the smallest scale value displayed on the vertical ruler.

**6. Vertical Ruler Maximum Value**

The maximum value of the vertical ruler refers to the highest scale value displayed on the vertical ruler.

**7. Horizontal Ruler Visibility**

Set the visibility of the horizontal ruler

**8. Vertical Ruler Visibility**

Toggle the visibility of the vertical ruler.

**9. Grid Display**

Set the visibility of the grid

**10. Background Color**

Set the background color.

**11. Zero Line Color**

Set the color of the X-axis and Y-axis zero tick marks

**12. Grid Line Color**

Set the color of the grid lines

**13. Dynamic Plotting Colors**

Invalid Property

**14. Proportionally**

Centering the coordinate origin and drawing scale marks according to a specified step size.

**15. Horizontal reversal**

Invert the direction of the horizontal axis.

**16. Vertical Inversion**

Invert the direction of the vertical axis.

## 6.27.4. Script Macros

"Details can be found in 'HUserGraphics_Designer_CN.xml'."

## 6.28. Calculator

## 6.29. Permission Management

## 6.29.1. Component Appearance

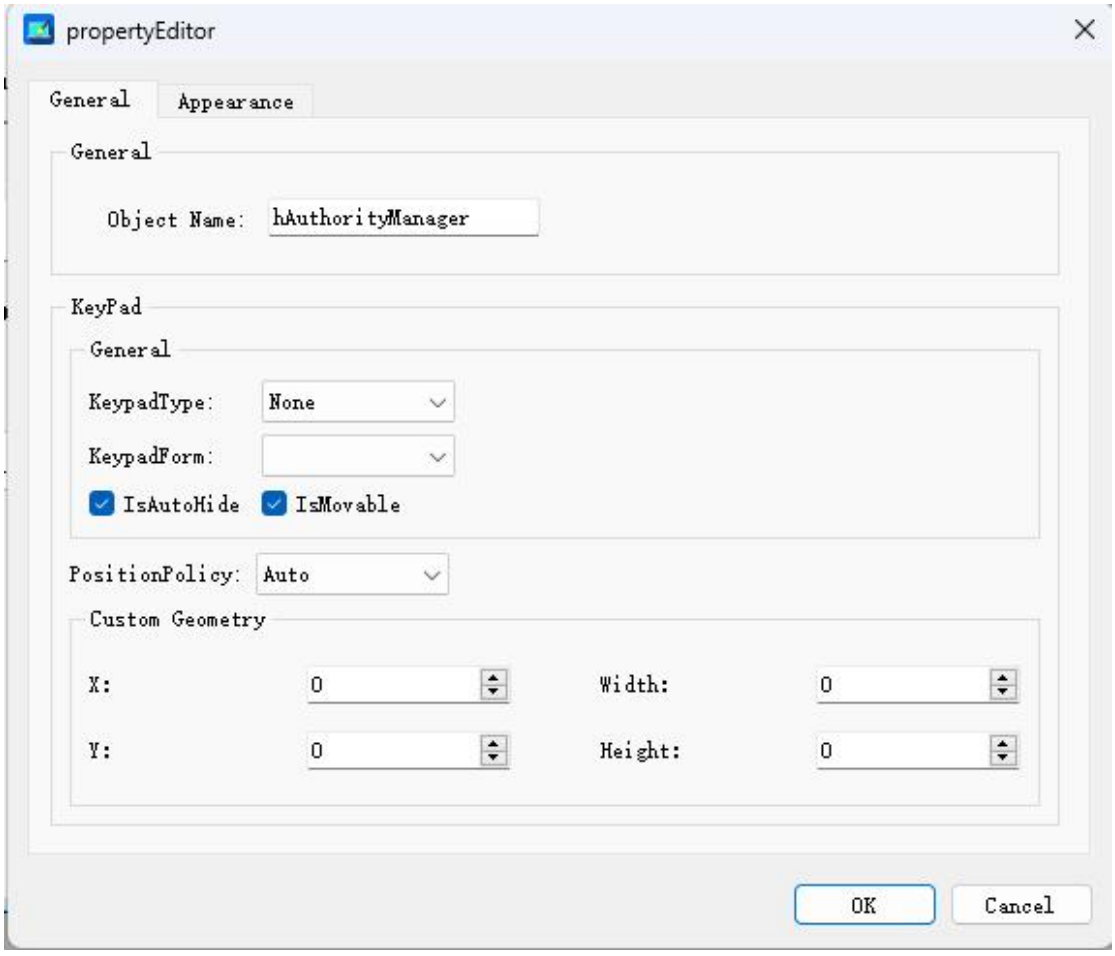
The component consists of a level selection dropdown, password input field, "remember password" checkbox, login button, cancel button, and reset password button.

## 6.29.2. Introduction

The permission management plugin provides functions for logging into HMI (Human-Machine Interface) and resetting permission passwords.

## 6.29.3. Property Editing Popup

The property editing popup window includes two tabs: "General Settings" and "Appearance."




### 1. General Settings

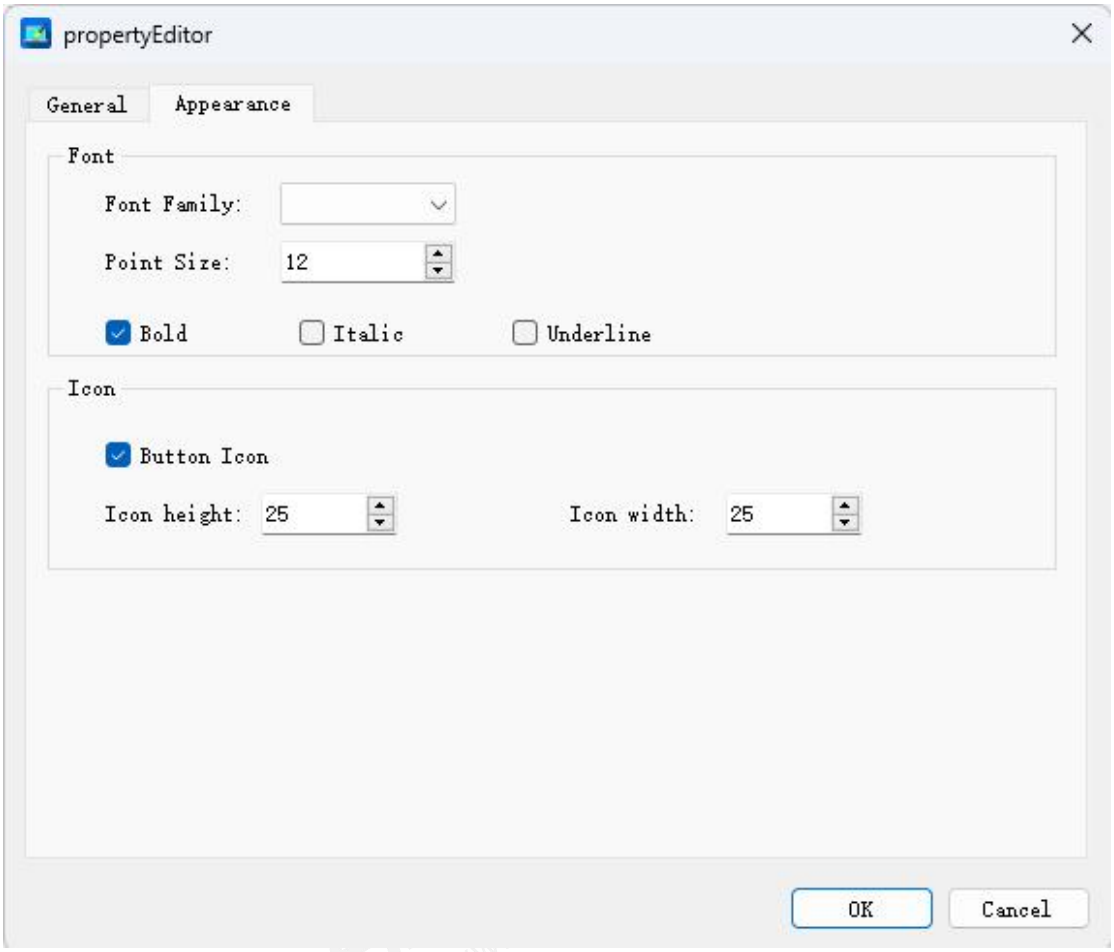
1) Object Name

Each object can be identified and accessed through its Object Name. The Object Name is a string used to reference and manipulate objects in script code.

**2. Virtual Keyboard**

When using touch input, a virtual keyboard can be used. By default, this is set to the password input keyboard. For details on the virtual keyboard

## 6.29.4. Appearance




**1. Font**

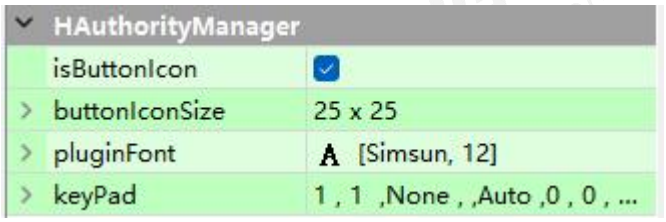
Plugin overall font settings

**2. Icons**

Plugin overall icon settings include icons for the login button, cancel button, and reset password button.

## 6.29.5. Property Editor

The attribute editor is positioned by default on the right side of the visual development software, where all attributes are reflected in the attribute editing popup window.




## 6.29.6. System Multilingual Support

Translations of inherent text on plugins, including file information display areas and plugin prompt texts, can be set in the "Tools" menu under "Component Inherent Text Translation."

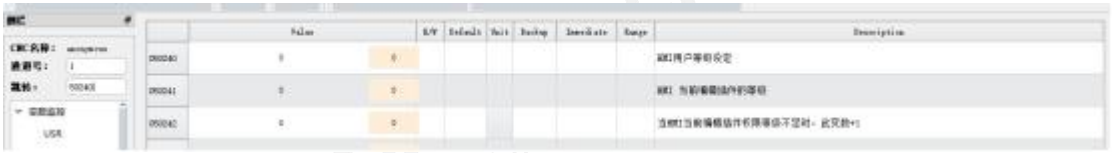

Please refer to the corresponding documentation for the inherent text translation editing

window functionality.

## 6.29.7. Function Details

### 6.29.7.1. Permissions

1. Input-type plugins (such as numerical plugins, code editors, tables, etc.) can be configured with operational permissions. By default, plugins have a permission level of 0, indicating no specific permissions set.
2. When the controller's human-machine interface (HMI) permission level is greater than the plugin's required permission level, access is insufficient. For example, if the plugin requires a permission level of 1 but the controller's permission level is 2, the access is insufficient to operate the plugin.
3. The variables related to permissions are as shown in the diagram:



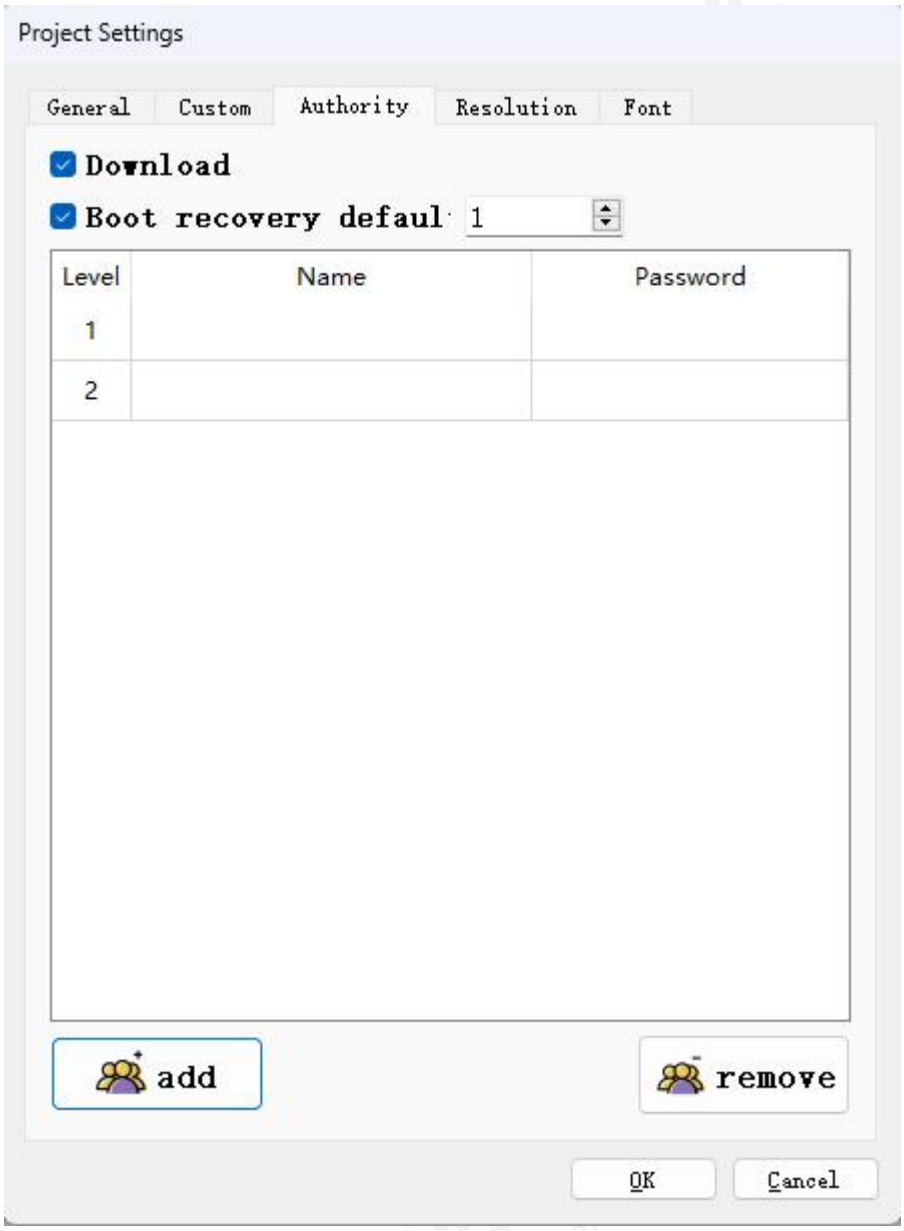
COM50240: Represents the HMI user level setting, specifically the current controller's human-machine interface (HMI) permission level.

COM50241: Represents the required permission level of the current focused component in the HMI editing plugin. For instance, if the focused component is a numerical plugin set with user permission level 2, then COM50241 variable value will be 2.

COM50242: When the current HMI editing plugin lacks sufficient permissions, the variable value increments. (The controller's HMI access level is 2, while the required permission level for the target plugin is 1. Editing the target plugin increases COM50242 by 1, prompting the need to log in with a higher permission level.) Application engineers will link this variable to trigger a permission login window.

## 6.29.8. Authority setting

The HMI man-machine permission can be set through the finger hmi designer software.On the Settings- -> Engineering Settings- -> Permissions Settings page.




1. "Reset the permission level and password", when updating the picture project, the permission setting information of the current screen project will be used to cover the permission information under the controller.If not checked, the permission setting under the previous controller will be maintained after updating the screen project.
2. Power recovery default level ". After checking, the controller will automatically adjust

the permission level to the target level, otherwise maintain the permission level before shutdown.

3. You can adjust the number of permissions by using "Add" and "Remove" buttons. Permission names support multi-language settings.

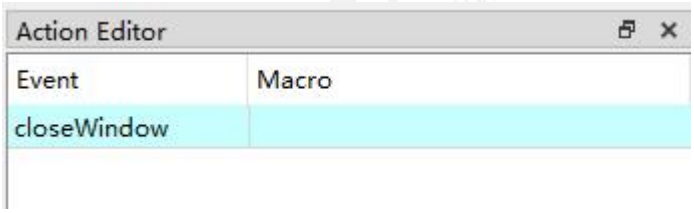
## 6.29.9. Login Permissions

Select the desired permission level for login, enter the password, and access the corresponding permission level.

## 6.29.10. Reset Password

To reset the password for a selected permission level, follow these steps:To reset the password, enter the original password and enter the new password twice to reset the password.

## 6.29.11. Close window

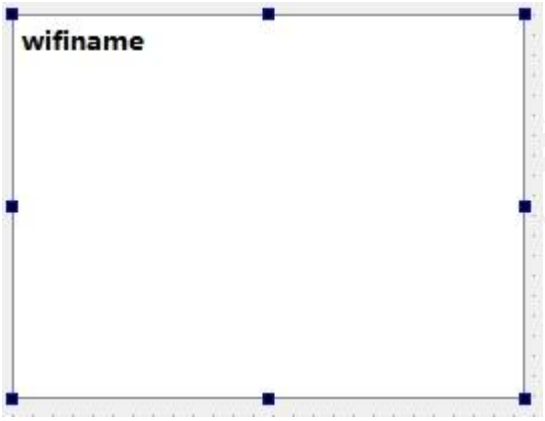



In the action editor, there's a "Close Window" event that sends a "closewindows" signal upon successful login, login cancellation, or successful password reset. This event is supposed to trigger the closure of the current window screen where the plugin is located. However, in practice, the HMI (Human-Machine Interface) does not currently support closing the main window screen.

## 6.30. Perpetual Calendar

# 6.31. Perpetual Calendar

## 6.31.1. Component Appearance




The plugin lists all scanned WiFi names in the form of a list.

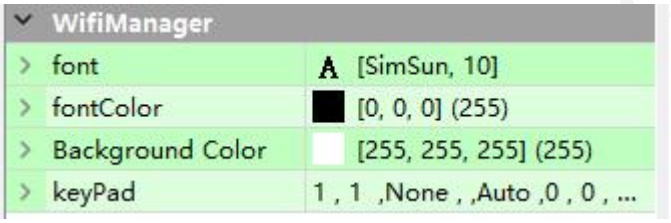
## 6.31.2. Introduction

The Wireless Management Plugin is primarily used for connecting to WiFi. It can only be used under the controller and requires a WiFi module for operation.

## 6.31.3. Property Editing Dialog

Plugin has no property editing dialog.

## 6.31.4. Property Editor




1. Font: sets the overall font for the plugin uniformly.
2. Font Color: uniformly sets the overall font color for the plugin.
3. Background Color: sets the background color for the plugin.

4. Virtual Keyboard: Plugin virtual keyboard settings

## 6.31.5. Detailed Description of Features

The Wireless Management Plugin provides functionalities such as scanning WiFi networks, connecting to WiFi, and disconnecting from WiFi. For details, please refer to the plugin's interface documentation.

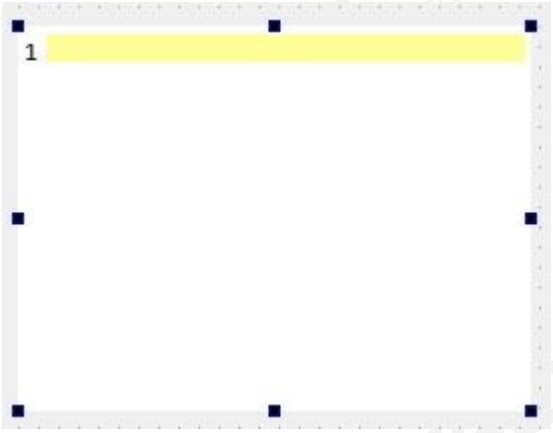
## 6.31.6. Plugin Function Table

| Function prototype | Description | Argument | Returned value |
|---|---|---|---|
| connectWifi() | Connect the selected WiFi, when triggered, it will determine if the WiFi that remembers the password is directly connected, otherwise the password input window will pop up | None | None |
| cursorDown() | Move the cursor down | None | None |
| cursorUp() | Move the cursor up | None | None |
| disconnect() | Disconnect The currently connected WiFi wireless card is in Inactive state. | None | None |
| getWifiInformation() | Get WiFi connected information | None | Array：List of strings that store WiFi information |

| | | | |
|---|---|---|---|
| | | | Inf[0] = Ssid (WiFi name)<br>Inf[1] = Status (Status)<br>Inf[2] = IP<br>Inf[3] = Encryption (encryption)<br>Inf[4] = Authentication (authentication)<br>Inf[5] = signal (signal strength)<br>Inf[6]= bssid |
| scan() | Scan the WiFi and display the connected WiFi to the main form. | None | None |

## 6.32. Code Display

### 6.32.1. The appearance of the component




The component consists of a line number display area and a text display area.

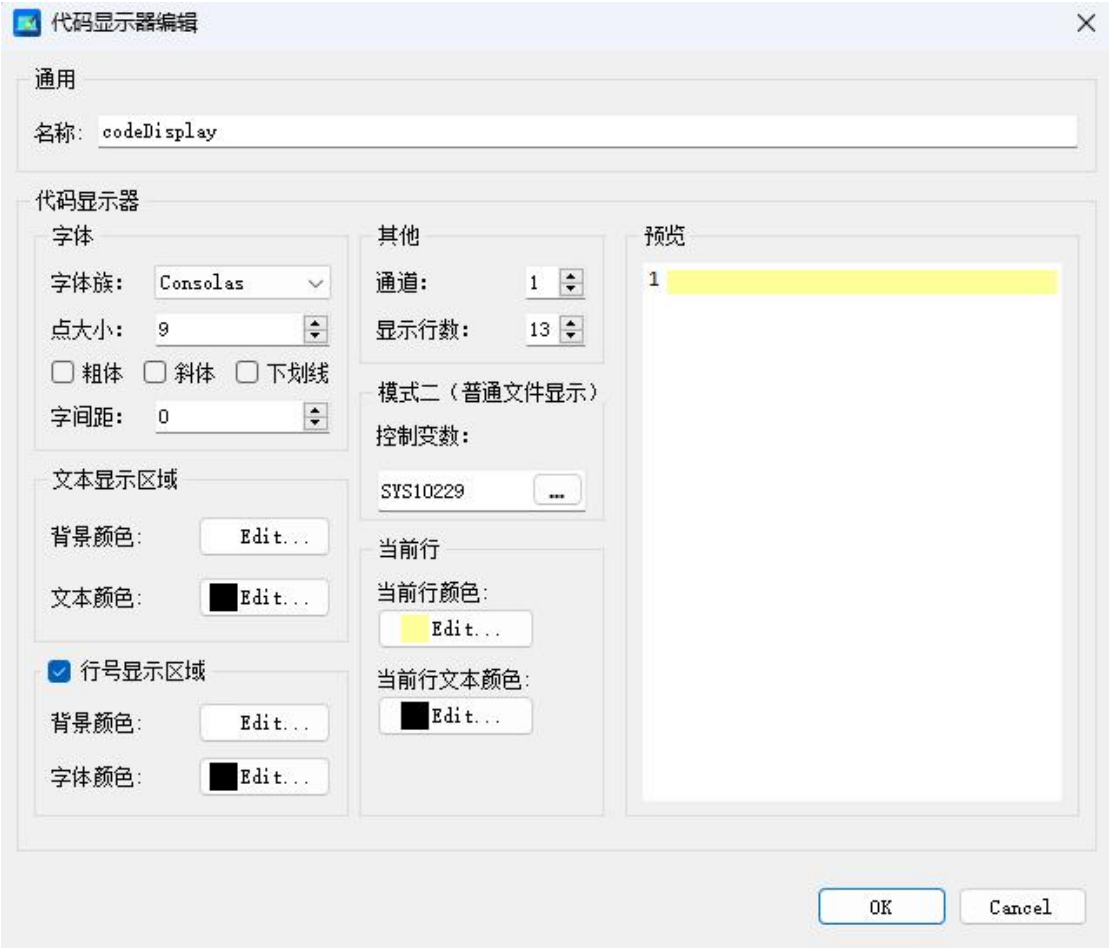
### 6.32.2. Introduction

The code display plugin is used to display program content loaded by the controller.The code display has three operating modes:

代码显示器的 3 个工作模式：

- The most commonly used default operating mode automatically loads the program file executed by the set channel. It refreshes the plugin viewport display based on SYS10228 (current execution position of the program) and SYS10229 (current execution line number of the program).
- Mode 1 displays simulated program files and automatically loads the currently simulated program file. It refreshes the plugin viewport display based on SYS10297 (current execution position of the program) and SYS10298 (current execution line number of the program).
- When enabling Mode Two via the function interface, it requires opening the target file through the interface and setting the control variables for Mode Two to refresh the plugin viewport.

## 6.32.3. Property Editing Dialog




**1. General**

1) Name

Every object can be identified and accessed through its Object Name. The Object Name is a string used to reference and manipulate objects in script code.

**2. Code display panel**

2) Font

Set the overall font properties for the plugin, including the text display area and the line number display area.

3) Text display area

a. Background color: Set the background color for the text display area

b. Text color: Set the text color for the text display area

4) Line number display area

a. You can control whether the line number display area is visible.

b. Background color: Set the background color for the line number display area.

c. Font color: Set the text color for the line number display area.

5) Other

a. hannel: Controls which program content loaded under which channel is displayed in the code editor.

b. Line number: Set the maximum number of lines visible in the code display viewport

6) Mode Two

a. Control variable: When the code display mode two is enabled, the current line is determined by the associated control variable.

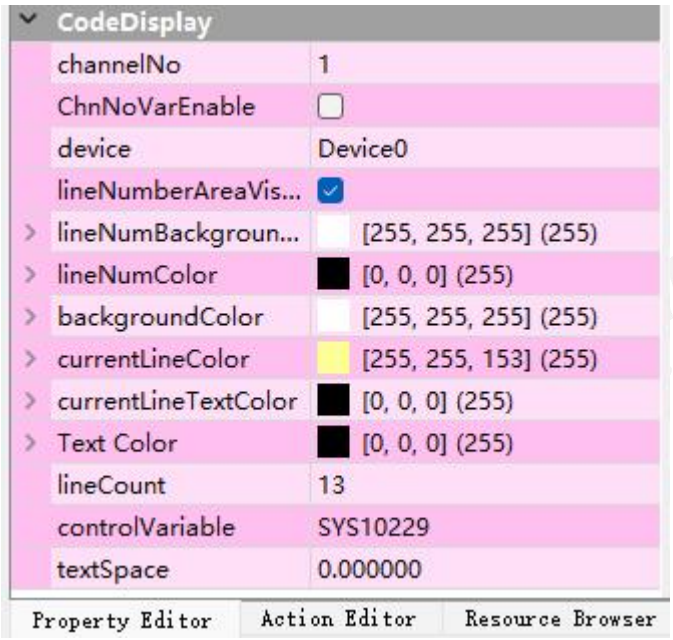
7) Current line

a. Current line color: The highlight color of the current line.

b. Current line text color: The color of the text in the current line

## 6.32.4. Property editor

The property editor defaults to the right side of the visual development software, and all properties are reflected in the property editing dialog.




## 6.32.5. Property Interface

| Attribute | Description | Usage |
|---|---|---|
| visible | Read and write plug-in visible properties | var visible = Form.codeDisplay.visible |
| enable | Read and write plug-in enable attributes | Var enable = Form.codeDisplay.enable |
| geomerty | To read and write the geometry properties of the plug-in, you need to use HProperty to read and write the specific value. | var geomerty = Form.codeDisplay.geomerty |
| font | To read and write the font properties of the plug-in, you need to use HProperty to read and write the specific value. | var font = Form.codeDisplay.font |
| background | To read and write the | var color = Form.codeDisplay. |

| | | |
|---|---|---|
| Color | background color of the code display, you need to use HProperty to read and write the specific value. | backgroundColor |
| channelNo | The channel number of the read/write code display | Form.codedisplay.channelNo=1 |
| ChnNoVarEnable | Sets whether the channel number variable control for the code display is valid | Form.codedisplay.ChnNoVarEnable = true |
| currentLineColor | Read/write code display current forward color | var color = Form.codeDisplay. currentLineColor |
| currentLineTextColor | Read/write code display current forward text color | var color = Form.codeDisplay. currentLineTextColor |
| lineCount | The number of lines of read/write code display | var count= Form.codeDisplay. lineCount |
| lineNumBackgroundColor | The background color of the line number field of the read/write code display | var color =Form.codedisplay. lineNumBackgroundColor |
| lineNumberAreaVisible | Whether the read/write line number area is visible | Form.codedisplay.lineNumberAreaVisible = true |
| lineNumColor | Read and write code display line number font colo | var color =Form.codedisplay.lineNumColor |
| txtColor | The font color of the read/write code display | var color =Form.codedisplay.txtColor |

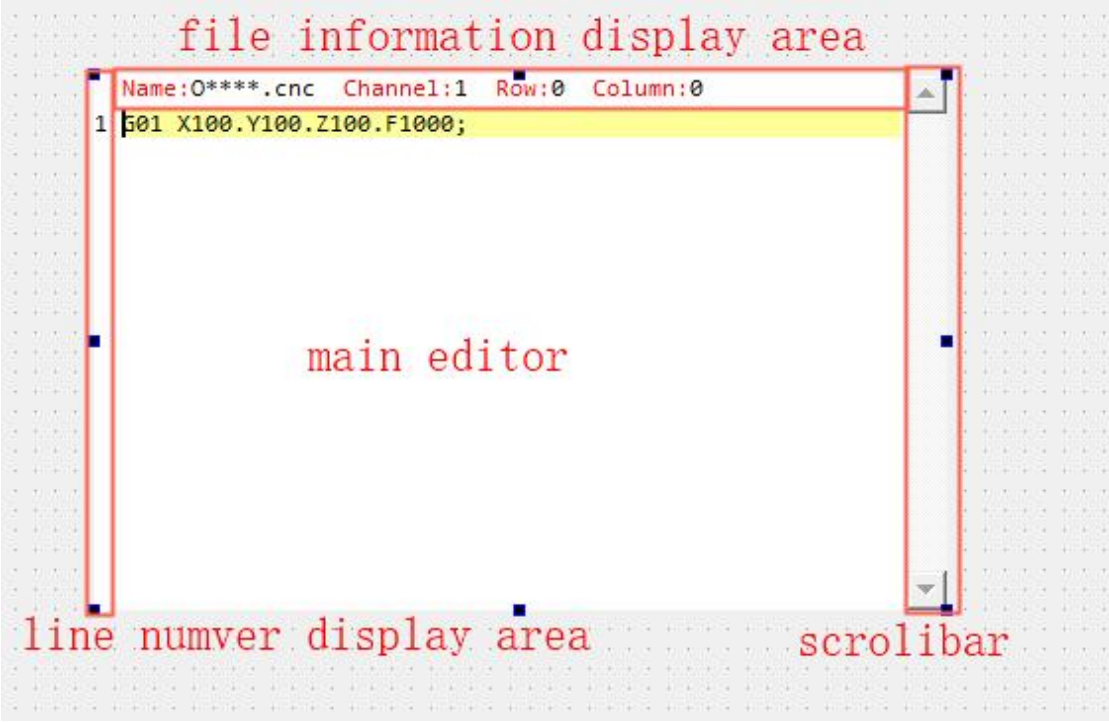
## 6.32.6. Interface functions

| Function prototype | Description | Argument | Returned value |
|---|---|---|---|

| | | | |
|---|---|---|---|
| void openFile(filepath) | Set the program file to open | (string)filePath: indicates the string file path. | None |
| void setDataMode(mode) | Sets the type of executable to display | int)mode[0-1], int indicates the mode type. The value can be 1 or 0. =0 Displays the actual execution program =1 Displays the simulation execution program =2 User-Defined | None |

## 6.33. Code editor

**Component appearance**



The component consists of a file information display area, a line number display area, the main editor body, and a scrollbar. The file information display area shows the file name, channel (determined by the file's path, e.g., "./usr/sys0001/program" corresponds to Channel 1), the current cursor line, and the current cursor column. The plugin can support up to three editors.

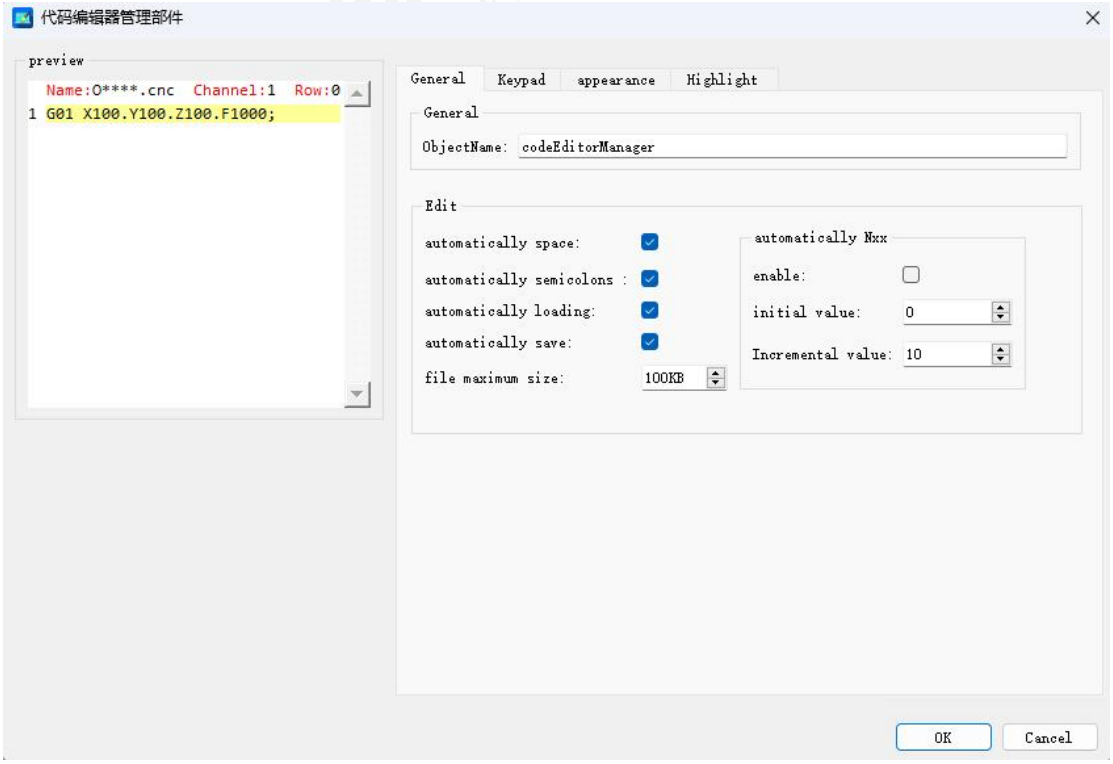
## 6.33.1. Introduction

The code editor is used for reading and writing files, supporting various formats, especially for editing program files (such as G-code).

## 6.33.2. Property Editing Dialog

The property editing dialog includes four tabs: "General," "Virtual Keyboard," "Appearance," and "Highlight.

[Universal]



## 1. Universal

1) Object Name

Each object can be identified and accessed by setting an object name. The object name is a string used to reference and manipulate objects in script code.

## 2. Auxiliary Editor

### 1) Automatically Add Interval

When editing G-code, a space is automatically added between two G-codes. For example, entering X100 followed by Y200 will display as X100 Y200.

### 2) Automatic Semicolon Addition

When editing G-code, a semicolon (;) is automatically added at the end of the line when pressing Enter.

### 3) Automatic Text Loading

When enabled, during the operation of the project, if the screen switches to the code editor view, the plugin will automatically load the recorded program file.

### 4) Automatic File Saving

When enabled, during the operation of the project, when switching from the editor view to other screens, the plugin will automatically save the current file.

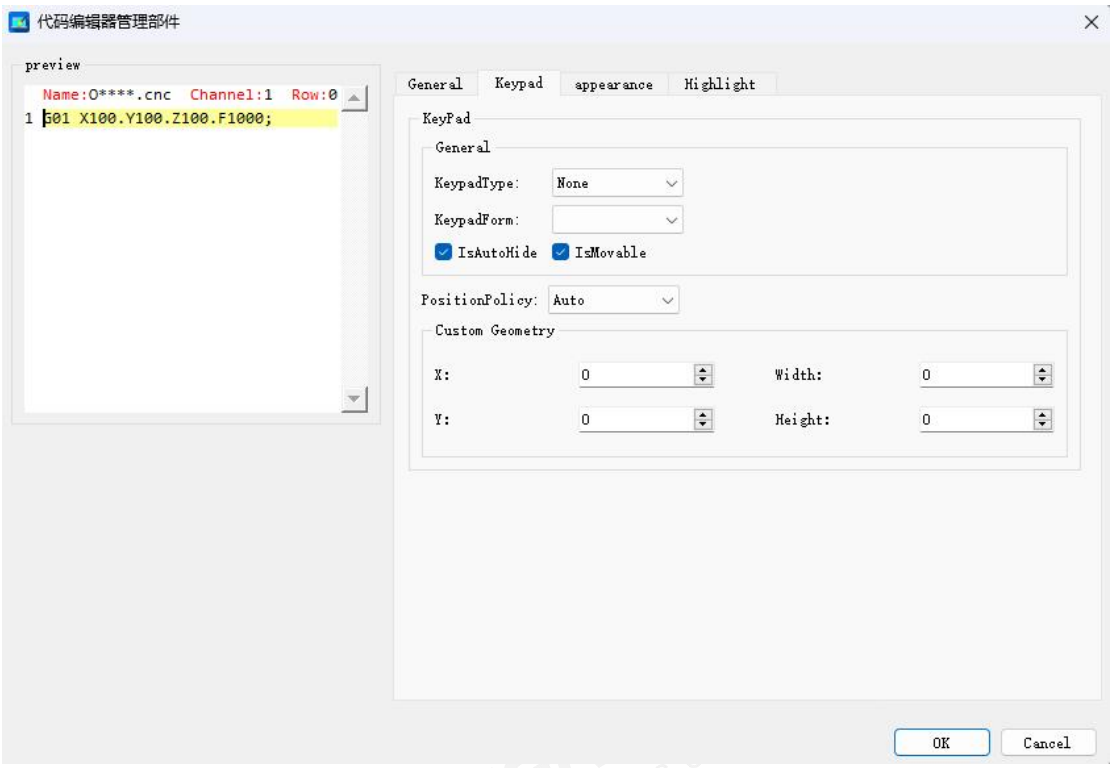
### 5) Maximum Size for Regular Files

The plugin classifies program files into "Large Files" and "Small Files" for better read/write handling. Files larger than the "Maximum Size for Regular Files" are considered large files; otherwise, they are small files.

### 6) Automatic Nxx Addition at Line Start (Automatic Numbering)

Attributes include "Automatic Numbering," "Automatic Numbering Start Value," and "Automatic Numbering Increment Value." When enabled, the editor will automatically insert an N code at the beginning of each line, with the N code value determined by the start value and increment value (start value + number of line breaks * increment value). This function can be controlled through attributes or functions.

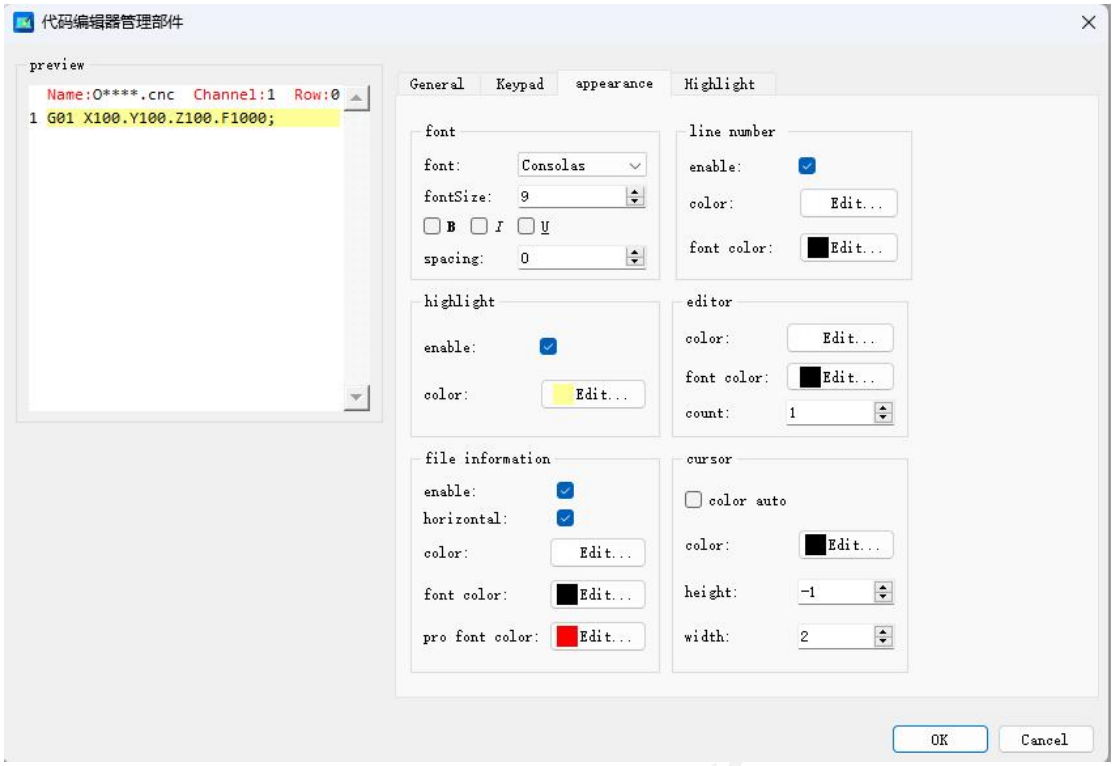
[Virtual keyboard]




(1) Virtual keyboard

When using touch input, you can use the virtual keyboard.

[Appearance]




**1. Font**

Unified font settings for plugin fonts, including file information display area, line number display area, and editor body font.

**2. Highlight the current row**

1) Is the control attribute enabled?

After enabling the attribute, the plugin will highlight the text block of the cursor's row with the "current row color".

2) Color (current row color)

Highlights the current row color.

**3. File information display area**

1) Enable

Control the visibility of the file information display area

2) Display horizontally

Enable the property to display file information entries horizontally; otherwise, display them vertically.

3) Color

Background color of the file information display area.

4) Font color

Font color for the file information display area.

5) Item font color

Font colors for items in the file information display area, including "Name," "Channel," "Line," and "Column."

**4. Line number display area**

1) Enable or not

Control whether the line number display area is visible.

2) Color

Background color for the line number display area.

3) Font Color

Font color for the line number display area.

**5. Text editor**

1) Color

Editor main background color

2) Font Color

Editor main background color

3) Editor Count

The plugin editor count specifies that the plugin can provide up to three editors.

**6. Cursor**

1) Automatically Invert Colors

Once enabled, the cursor color will automatically invert based on the background color of the text block it resides in, and the cursor color cannot be manually set.

2) Color

Set the cursor color. Once "Automatically Invert Colors" is activated, the property becomes uneditable.

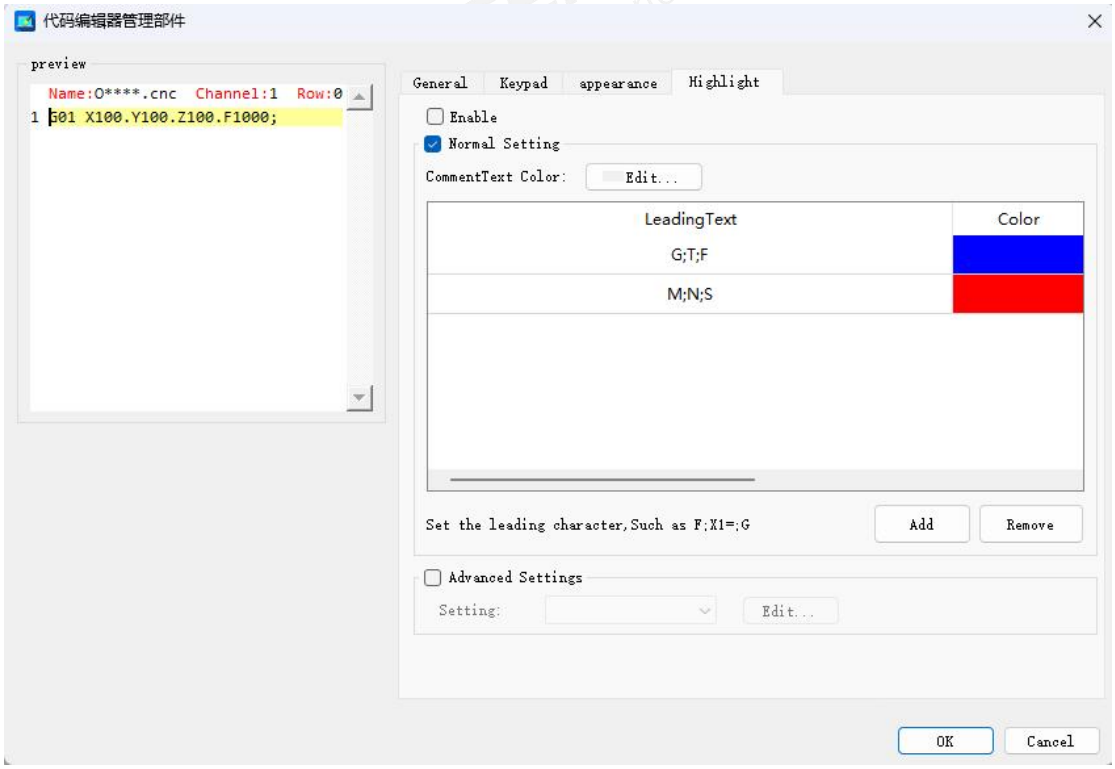
3) Height

You can set the cursor height, with a default value of -1,which automatically adjusts based on the font size.

4) Width

You can set the width of the cursor, with a default value of 2.

[Highlight]




**1. Enable**

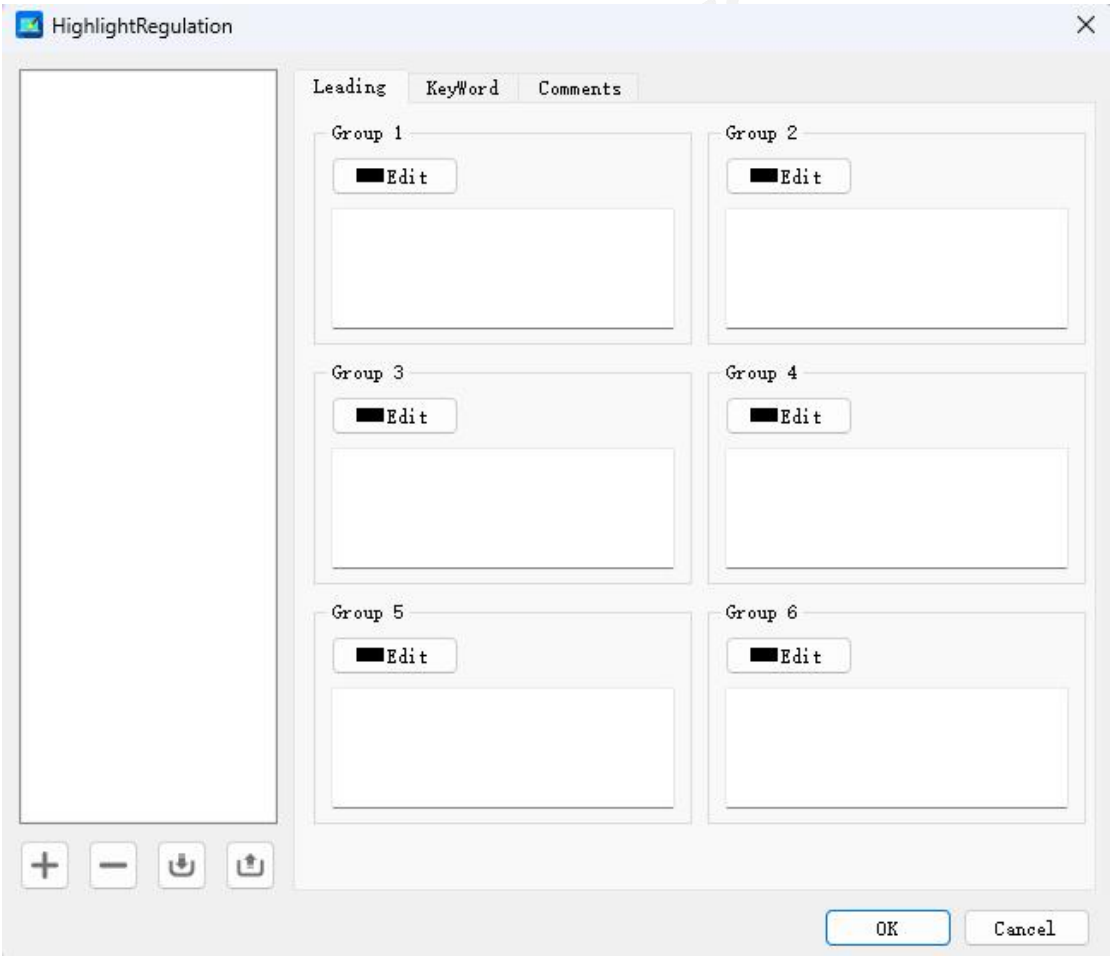
Used to enable or disable the highlighting functionality. There are two types of highlighting features: basic and advanced. These cannot be enabled simultaneously. When using the basic highlighting feature, the advanced highlighting feature is disabled, and vice versa.

1) Basic

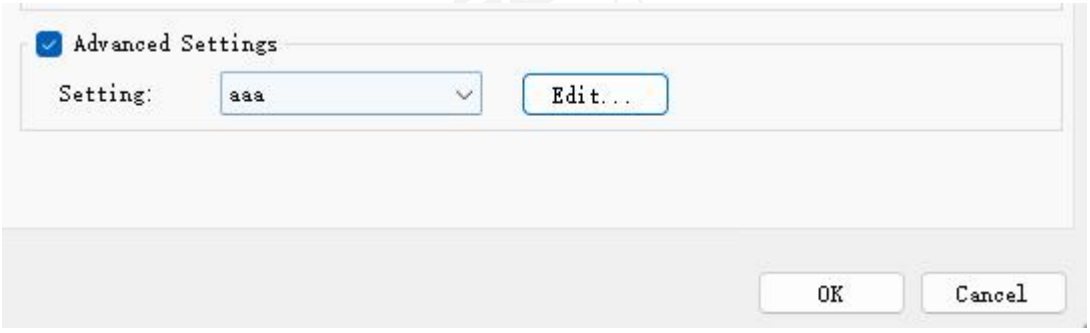
The basic highlighting functionality allows editing settings within the property editing dialog. This includes configuring colors for comments and lead text (G-codes), where different G-codes are separated by semicolons ";". You can add or remove highlight colors as needed.

2) Advanced

Primarily used for complex and specific text highlighting requirements, enabling the "Advanced" checkbox activates advanced functionality. Clicking "Edit" opens a popup window for editing advanced highlight settings, as shown in the following figure:




Allow creation, deletion, export, and import of highlight configurations. Highlight configurations include options for prefixes, keywords, and comments. Once edited, these configurations can be selected in the "Highlight" page of the property editing dialog in the code editor, as shown in the following figure:



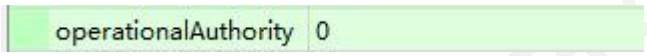
## 6.33.3. Property Editor

The default position of the property editor in the visual development software is on the right side. Most properties are reflected in the "Property Editing Dialog"; therefore, I won't list them all here. The following text will focus solely on properties not included in the editing dialog.

**1. Embedded Keyboard**


When the software keyboard is enabled, this controls whether the software keyboard is embedded within the plugin. The embedded keyboard cannot be moved and is the same width as the plugin, with a height equal to half of the plugin's height.

**2. Operational Permissions**




The default value is 0, indicating no permission restrictions are enabled. This setting establishes operational permissions for the component. If permissions are insufficient, the plugin will be unable to edit. (Refer to permission instructions.)

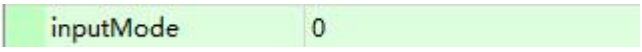
3. **Slider Width or Height**

Used to control the width or height of the slider in the plugin.

**4. Read-Only**

The properties involved are "Read-Only" and "Read-Only Control Bit". When the read-only property is enabled, the editor is in a non-editable state. You can associate the read-only control bit with a system address to control the plugin's read-only attribute through that system address.

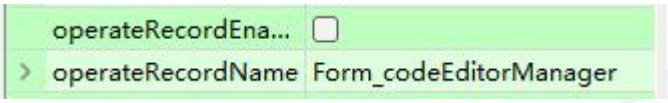
**5. Input Mode (Whether to Use GSK Block Input Mode)**




When the input mode is set to 1, the plugin is in "GSK block input mode". In this mode, direct input or soft keyboard input is not possible. Instead, you need to use the " GSK

input block" in conjunction with the Insert and Delete keys to perform operations such as adding, deleting, or modifying text within the editor.

**6. Operation Log**




1) Set whether to enable operation recording.
2) Set the plugin name on the operation record, with a default value of "画面名_插件对象名".

To view the specific functionality of the operation record, please refer to the corresponding documentation for details.

## 6.33.4. System Multilingual Support

The translation of inherent text within the plugin, including file information display areas and plugin tooltip text, can be configured under the "Tools" menu in the "Component Inherent Text Translation" section.


Inherent Text Translation Editing Window Functionality - Please refer to the corresponding documentation for details.

## 6.33.5. Detailed Feature Description

The plugin's functionality extends beyond the attributes mentioned above. The following sections will detail each feature of the code editor plugin comprehensively.

[ Jump]

The code editor provides multiple interfaces for implementing navigation functionalities. The first method involves invoking the jump window using the interface function openJumpDialog, as illustrated in the following figure:


The second method involves setting the target line via a jump window and triggering the jump using the interface function jumpTo.

[Search and Replace]

The plugin provides multiple interfaces to achieve the functionalities of searching, replacing, and replacing all. The first method involves invoking the search and replace

window using the interface function openFindDialog, as illustrated in the following figure:


"Set the find and replace content along with search criteria including reverse search, case sensitivity, and whole word matching. Implement functionalities to trigger find, replace, and replace all operations. Alternatively, utilize interface functions such as setFindString or setReplaceString to specify the content for finding or replacing, then trigger the editorReplace function."

[Load Content to MDI]

The plugin provides the interface function "loadToMdi" to enable writing the editor content to the controller's MDI (Manual Data Input) functionality.

[Program Restart]

The plugin uses the interface function "setRestartFromHere" to instruct the controller to load the current program and execute it starting from the line where the cursor is located.

[Highlight Changed Lines]

The plugin provides the "setHighlightChangedLine" interface to enable highlighting of modified lines. Once this feature is enabled, the "Highlight Current Line" feature will be automatically disabled. When the text content is modified, the plugin will highlight the modified line with a color specified for "current line highlighting". After saving the file, the highlight will disappear. Note that this feature is only available for "small files".

[Code Help]

The plugin provides an interface to enable G-code help. When the interface is called to open the code help window, the editor displays corresponding content based on the current cursor's position on the G-code in the help window. By default, there are two types of code help: lathe and milling. The system help documents are stored in "./system/config/help/". Please ensure that there are help documents available in the default path for the help functionality to work properly.

[Program Restart]

The interface setRestartFromHere loads the current file in the editor and starts execution from the current line.

## 6.33.6. Plugin Properties

| Attribute | Description | |
|---|---|---|
| autoAddN | The read-write code editor automatically adds N-code property values | codeEditorManager.autoAddN=true |
| autoLoad | The read-write code editor automatically loads property values | codeEditorManager.autoLoad=true |

| | | |
|---|---|---|
| autoSave | The read-write code editor automatically saves properties | codeEditorManager.autoSave=true |
| autoSemicolon | The read-write code editor automatically adds semicolon attributes | codeEditorManager.autoSemicolon=true |
| autoSpace | The read-write code editor controls automatically add properties | codeEditorManager.autoSpace=true |
| backgroundColor | Read and write code editor background color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.backgroundColor<br>=HProperty.setQColor(colorMap) |
| currentLineColor | Read/write code editor current line color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.currentLineColor<br>=HProperty.setQColor(colorMap) |
| cursorColor | Read and write code editor cursor color | var colorMap={};<br>colorMap["red"]=255; |

| | | |
|---|---|---|
| | | colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.cursorColor<br>=HProperty.setQColor(colorMap) |
| cursorHeight | Read and write code editor cursor height | codeEditorManager.cursorHeight=10 |
| cursorWidth | Read and write code editor cursor width | codeEditorManager.cursorWidth=10 |
| editorNum | Gets the number of editors for the code editor control, properties read-only | num = codeEditorManager.editorNum |
| fileInfBackground Color | Background color for reading and writing code editor file information | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.fileInfBackgroun dColor<br>=HProperty.setQColor(colorMap) |
| fileInfHFormat | Whether the read-write code editor displays file information horizontally | codeEditorManager.fileInfHFormat=true |
| fileInfItemColor | Read and write code editor file information Project field text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.fileInfItemColor<br>=HProperty.setQColor(colorMap) |

| | | |
|---|---|---|
| fileInfTxtColor | Text color for reading and writing code editor file information | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.fileInfTxtColor<br>=HProperty.setQColor(colorMap) |
| fileTxtColor | Read and write code editor text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.fileTxtColor<br>=HProperty.setQColor(colorMap) |
| highlightCurrentLine | Whether the read/write code editor highlights the current line | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.highlightCurrentLine<br>=HProperty.setQColor(colorMap) |
| inputMode | Read and write code editor input mode | codeEditorManager.inputMode=1 |
| insetSoftKeypad | Whether the read and write code editor is embedded in the software disk | codeEditorManager.insetSoftKeypad=true |
| lineNum | Whether the read/write code editor displays line numbers | codeEditorManager.lineNum=true |

| | | |
|---|---|---|
| lineNumBackgroundColor | Read and write code editor line number field background color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.lineNumBackgroundColor<br>=HProperty.setQColor(colorMap) |
| lineNumColor | Read and write code editor line number field text color | var colorMap={};<br>colorMap["red"]=255;<br>colorMap["green"]=0;<br>colorMap["blue"]=0;<br>Form.codeEditorManager.lineNumColor<br>=HProperty.setQColor(colorMap) |
| normalFileMaxSize | Read and write code editor common file size limits | codeEditorManager.normalFileMaxSize=102400 |
| openFileInf | Whether the read/write code editor displays file information | codeEditorManager.openFileInf=true |
| readOnly | Whether the read-write code editor is read-only | codeEditorManager.readOnly=false |
| scrollBarExtent | Read and write code editor slider size | codeEditorManager.scrollBarExtent=25 |
| softKeypad | Whether the read and write code editor uses a soft keyboard | codeEditorManager.softKeypad=true |
| softKeypadAutoHide | Whether the read and write code editor is | codeEditorManager.softKeypadAutoHide=true |

| | | |
|---|---|---|
| | automatically hidden | |

## 6.33.7. Plug-in function table

| Function prototype | Description | Argument | Returned value |
|---|---|---|---|
| activeDeleteOneLine() | Delete the line where the cursor is located in the currently active editor. | None | None |
| activeEditorID() | Get the ID of the currently active editor. | None | ID of the currently active editor (0-2) |
| activeInsertText(text) | Insert text into the currently active editor. | (string) text content | None |
| activeJumpTo(line) | Jump to the Nth line in the currently active editor. | (int) target line | (bool) Whether the file opened successfully |
| activeOpenFile(fileName ,type,device) | Open a file in the currently active editor. | (string) fileName: file path, string. (int) type [0-1]: file type = 0 normal CNC file; = 1 CNC file with file information; this parameter can be omitted. (string) device: device name, | (bool) Whether the file saved successfully |

| | | | |
|---|---|---|---|
| | | default parameter "Device0" | |
| activeSaveFile(fileName,device) | Save the file opened in the currently active editor. | (string) fileName: file path, string. (string) device: device name, default parameter "Device0" | None |
| chageFindDlgFocus() | Switch input focus in the find and replace dialog box. | None | None |
| changeJmpDlgFocus() | Switch the input focus in the jump dialog box. Since there is only one input box in this dialog, the macro will make the input box gain focus. | None | None |
| clearInputWidget() | Clear the input box, which can only be used in input mode 1. | None | None |
| clickFindBtn() | Click the Find button in the find and replace dialog box. | None | None |
| clickJmpDlgCancelBtn() | Click the Cancel button in the jump dialog box to | None | None |

| | | | |
|---|---|---|---|
| | close the dialog box and cancel the jump. | | |
| clickJmpDlgOKBtn() | Click the OK button in the jump dialog box to execute the jump function in the currently active dialog. | None | None |
| clickReplaceAllBtn() | Click the Replace All button in the find and replace dialog box. | None | None |
| clickReplaceBtn() | Click the Replace button in the find and replace dialog box. | None | None |
| closeAssistantDialog() | Close the help window. | None | None |
| closeFindDialog() | Close the find and replace dialog box. | None | Empty |
| confirmModify() | In the background editing function, call this function to confirm the modification and save it. The next execution will adopt the newly modified content. | None | None |
| deleteLine(line,ID) | Delete the line content in the specified editor. | (int) line: specified line (int) ID [0-2]: editor identifier | (string) Returned copied text content |
| deleteOneLine(ID) | Delete the line where the | (int) ID [0-2]: | None |

| | | | |
|---|---|---|---|
| | cursor is located in the specified editor. | editor identifier | |
| editorCopy() | Copy the selected text. | None | (bool) Whether the target string was found |
| editorCut() | Cut the selected text. | None | (bool) Whether it is hidden |
| editorFind(findBackward, caseSensitively,wholeWords) | Find the next occurrence. | (bool) findBackward: whether to search backward = false, forward search; = true, backward search. (bool) caseSensitively : whether to distinguish case = false, case insensitive; = true, case sensitive. (bool) wholeWords: whether to | None |

| | | | |
|---|---|---|---|
| | | match whole words = false, partial word match; = true, whole word match. | |
| editorIsHide(ID) | Get the display/hidden status of the specified editor. | (int) ID [0-2]: editor identifier | None |
| editorMoveCursor(mode) | Move the cursor position in the currently active editor. | (int) mode: movement mode, specifically: = 0, move to the end of the line; = 1, move to the beginning of the line; = 2, move to the beginning of the document; = 3, move to the end of the document; = others, move to the end of the line; | None |
| editorPaste() | Paste the selected text. | None | (bool) Whether the |

| | | | |
|---|---|---|---|
| | | | target string was found |
| editorRedo() | Redo the operation. | None | None |
| editorReplace(findBackward, caseSensitively,wholeWords) | Replace the next occurrence. | (bool) findBackward: whether to search backward = false, forward search; = true, backward search. (bool) caseSensitively : whether to distinguish case = false, case insensitive; = true, case sensitive. (bool) wholeWords: whether to match whole words = false, partial word match; = true, | (bool) Whether the auto-add-N function is enabled when wrapping |

| | | | |
|---|---|---|---|
| | | whole word match. | |
| editorUndo() | Undo the operation. | None | (int) Increment value in the auto-add-N function |
| getActiveIsAddN() | Check if the currently active editor has line wrapping with auto N addition enabled. | None | (int) Starting value of N in the auto-add-N function |
| getActiveNoffset() | Get the increment value in the auto N addition function of the currently active editor. | None | (string) File path |
| getActiveNstart() | Get the starting value of N in the auto N addition function of the currently active editor. | None | (int) Current line number of the editor |
| getActiveOpendFileName() | Get the file path of the file opened in the currently active editor. | None | (string) Current line text of the editor |
| getCurrentLineNo() | Get the current line number in the editor. | None | (string) String in the find input box of the find and replace dialog |
| getCurrentLineText() | Get the content of the current line in the editor. | Empty | (bool) Whether the auto-add-N function is enabled when wrapping |

| | | | |
|---|---|---|---|
| getFindString() | Get the string in the find input box of the find and replace dialog. | None | (int) Increment value in the auto-add-N function |
| getIsAddN(ID) | Check if line wrapping with auto N addition is enabled for the specified editor. | (int) ID [0-2]: editor identifier | (int) Starting value of N in the auto-add-N function |
| getNoffset(ID) | Get the increment value in the auto N addition function of the specified editor. | (int) ID [0-2]: editor identifier. | (string) File path |
| getNstart(ID) | Get the starting value of N in the auto N addition function of the currently active editor. | (int) ID [0-2]: editor identifier | (int) authority: Plugin restriction permissions |
| getOpendFileName(ID) | Get the file path of the file opened in the specified editor. | (int) ID [0-2]: editor identifier | (string) String in the replace input box of the find and replace dialog |
| getOperationalAuthority() | Get the plugin permission restriction. | Empty | (string) Obtained text content |
| getReplaceString() | Get the string in the replace input box of the find and replace dialog. | None | Empty |
| getTxt() | Retrieve the text content from the specified editor by ID. | (int) ID: editor number, range 0~2 | Empty |
| inputCode() | Input code into the editor, only in input mode 1. | Empty | None |

| | | | |
|---|---|---|---|
| insertCode() | Insert code into the editor, only in input mode 1. | Empty | None |
| insertLine(line , content , ID) | Insert a line of text into the specified editor. | (int) line: specified line (string) content: text to be inserted (int) ID [0-2]: editor identifier | (bool) Whether the content was modified |
| insertText(ID,text) | Insert text into the specified editor. | (int) ID [0-2]: editor identifier (string) text: inserted text content | None |
| isTextEdited(ID) | Check if the specified editor has been edited. | (int) ID: editor number, range 0~2 | None |
| jumpTo(ID,n) | Jump to the Nth line in the specified editor. | (int) ID [0-2]: editor identifier (int) n: target line, integer | (bool) Whether the file opened successfully |
| openAssistantDialog() | Open the help window. | None | None |
| openFile(ID,fileName,type,device) | Open a file in the specified editor. | (int) ID [0-2]: editor identifier (string) fileName: file path, string. | None |

| | | | |
|---|---|---|---|
| | | (int) type [0-1]: file type = 0 normal CNC file; = 1 CNC file with file information; this parameter can be omitted. (string) device: device name, default parameter "Device0" | |
| openFindDialog() | Open the find and replace dialog box. | None | None |
| openJumpDialog() | Open the jump dialog box. | None | None |
| overwriteLine(line , content , ID) | Overwrite the content of the specified line in the editor. | (int) line: specified line (string) content: text to be inserted (int) ID [0-2]: editor identifier | (string) Line content |
| pageDown() | Scroll down the page. | None | (bool) Whether the file saved successfully |
| pageUp() | Scroll up the page. | None | None |
| readLine(line , ID) | Get the text of line X in the | (int) line: | None |

| | | | |
|---|---|---|---|
| | specified editor. | specified line<br>(int) ID [0-2]:<br>editor identifier | |
| saveFile(ID,fileName) | Save the file opened in the specified editor. | (int) ID [0-2]:<br>editor identifier<br>(string)<br>fileName: file path, string.<br>(string) device:<br>device name, default parameter "Device0" | None |
| setActiveIsAddN(open) | Enable or disable line wrapping with auto N addition in the currently active editor. | (bool) open:<br>whether to enable the auto-add-N function. | None |
| setActiveNoffset(value) | Set the increment value in the auto N addition function of the currently active editor. | (int) value:<br>increment value in the auto-add-N function,<br>integer | None |
| setActiveNstart(value) | Set the starting value of N in the auto N addition function of the currently active editor. | (int) value:<br>starting value of N in the auto-add-N | None |

| | | | |
|---|---|---|---|
| | | function | |
| setDragMoveEnble(enable) | Allow or disallow text dragging. | (bool) enable: whether to allow text dragging | None |
| setEditorFocus(ID) | Transfer focus to the specified editor. | (int) ID [0-2]: editor identifier | None |
| setEditorMangerFocus() | Transfer focus to the editor manager, which decides which editor should get the focus. | None | None |
| setEditorSelectMode(open) | Enable or disable selection mode in the currently active editor. | (bool) open, whether to enable selection mode | None |
| setFindString(str) | Set the string in the find input box of the find and replace dialog. | (string) String in the replace input box of the find and replace dialog. | None |
| setHighlightChangedLine(enable) | Highlight lines of text content that have been modified. | (bool) enable: whether to highlight the modified lines | None |
| setIsAddN(ID,open) | Enable or disable line wrapping with auto N addition in the specified editor. | (int) ID [0-2]: editor identifier | None |

| | | | |
|---|---|---|---|
| setLineSelectMode(enable) | Enable or disable line selection mode. | (bool) open: whether to enable the auto-add-N function. true: automatically add N when wrapping; false: do not automatically add N when wrapping. | None |
| setNoffset(ID,value) | Set the increment value in the auto N addition function of the specified editor. | (int) ID [0-2]: editor identifier (int) value: increment value in the auto-add-N function | None |
| setNstart(ID,value) | Set the starting value of N in the auto N addition function of the specified editor. | (int) ID [0-2]: editor identifier (int) value: starting value of N in the auto-add-N function | None |
| setOperationalAuthority(authority) | Set plugin permission restrictions. | (int) authority: plugin | None |

| | | | |
|---|---|---|---|
| | | restriction permissions | |
| setReplaceString() | Set the string in the replace input box of the find and replace dialog. | (string) String in the replace input box of the find and replace dialog. | ID of the currently active editor (0-2) |
| setTxt(ID,text) | Set the text content for the specified editor by ID. | (int) ID [0-2]: editor number (string) text: string type, set text content; | None |
| showHideEditor(ID,show) | Show or hide the specified editor. | (int) ID [0-2]: editor identifier (bool) show: whether to display, if this parameter is omitted, the default is true | (bool) Whether the file opened successfully |
| toggleCaseSensitively() | Toggle the FindCaseSensitively option in the find and replace dialog box. When selected, case sensitivity is enabled. | None | (bool) Whether the file saved successfully |
| toggleEditorFocus() | When multiple editors exist, use this to switch the | None | None |

| | | | |
|---|---|---|---|
| | currently active editor. | | |
| toggleSearchDirect() | Toggle the FindBackward option in the find and replace dialog box. When selected, reverse search is enabled. | None | None |
| toggleWholeWords() | Toggle the FindWholeWords option in the find and replace dialog box. When selected, whole word matching is enabled. | None | None |
| setRestartFromHere() | Restart the execution of the program file from the current line. | None | None |

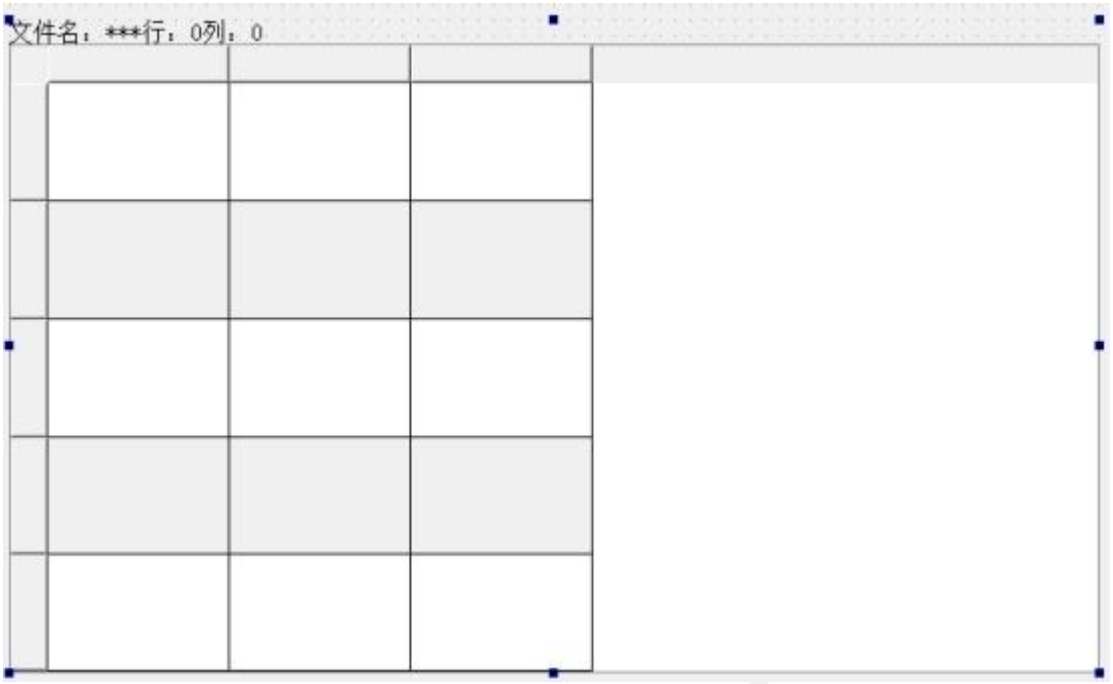
## 6.34. Text Box

## 6.35. Multi-line Text Box

## 6.36. Dropdown List

## 6.37. Table Set

### 6.37.1. Component Appearance




The components include the filename, row and column displays, and the table used to display the loaded program file contents.

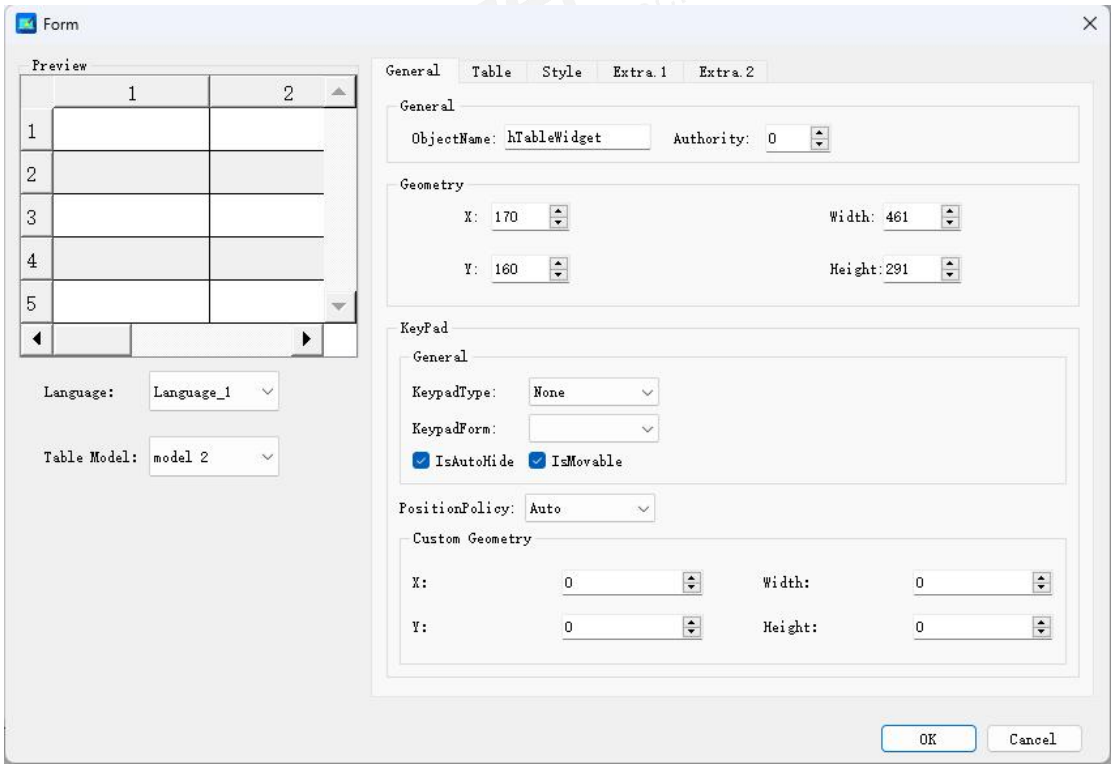
### 6.37.2. Introduction

The Table Set currently has 5 modes:Mode 1: Tool Compensation Mode;Mode 2: Fill Cell Mode;Mode 3: File Read/Write Mode;Mode 4: Custom Mode;Mode 5: IoT Mode

Mode 2, Fill Cell Mode, is capable of loading and displaying program file contents.

## 6.37.3. Property Editing Dialog

Property Editing Dialog includes Basic, Table, Style, Special.1, and Special.2 tabs. Common properties for each mode in the Table Set plugin are found in the Basic, Table, and Style tabs of the property dialog. Mode-specific properties are located in the Special tabs.




**1. Basics**

1) Object Name: Each object can be identified and accessed by setting its object name. The object name is a string used to reference and manipulate objects in script code.

2) Access Permissions: Set the operational permissions for the plugin.

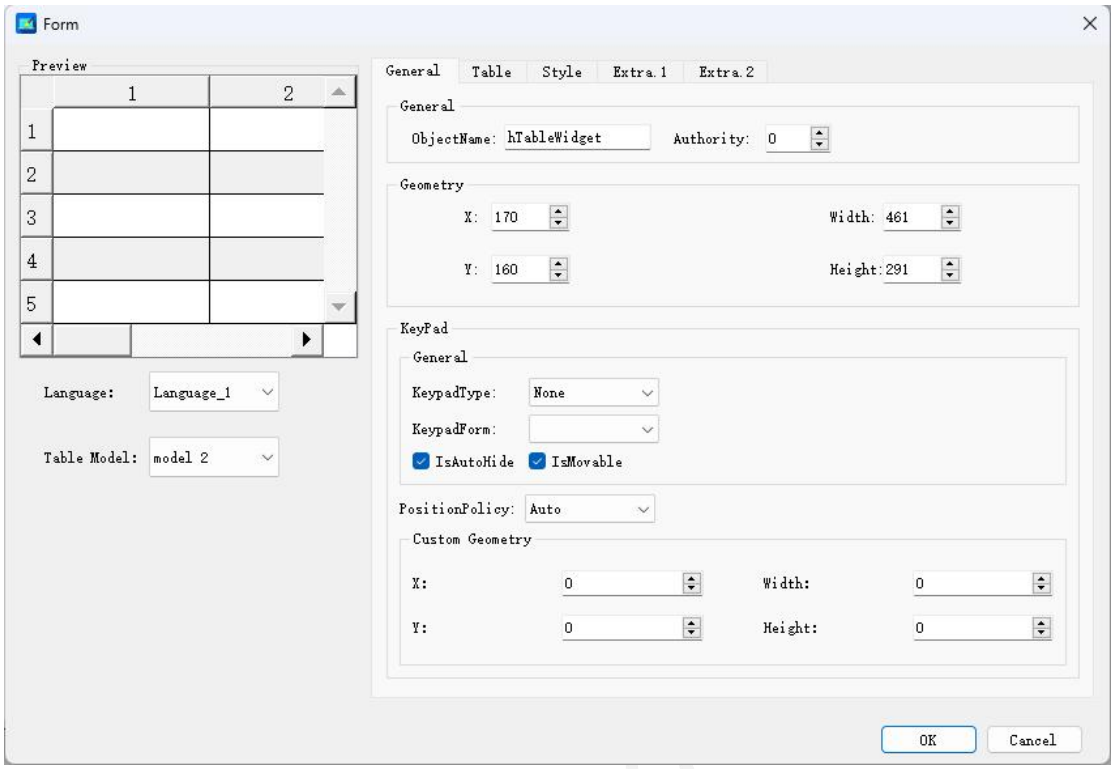
**2. Geometry Dimensions**

Properties for setting the position and size of the plugin.

**3. Virtual Keyboard**

When using touch input, you can use the virtual keyboard. Refer to the keyboard instructions for details. In Table Set Mode 2, the default virtual keyboard is the decimal

keyboard.




**1. Table**

1) Invisible Bit: Controls the visibility of the table set plugin, can be linked to system bit variables.

2) Viewport Columns: Set the number of columns visible in the table viewport. The viewport column count equals the actual number of columns in the table.

3) Viewport Rows: Set the number of rows visible in the table viewport. The viewport row count may differ from the actual number of rows in the table.

**2. Properties**

1) Fill Last Column: Automatically adjusts the width of the last column to fill the table.

2) Equal Column Width: Distributes the column width evenly based on the current table width, vertical header width, slider width, and the number of columns.

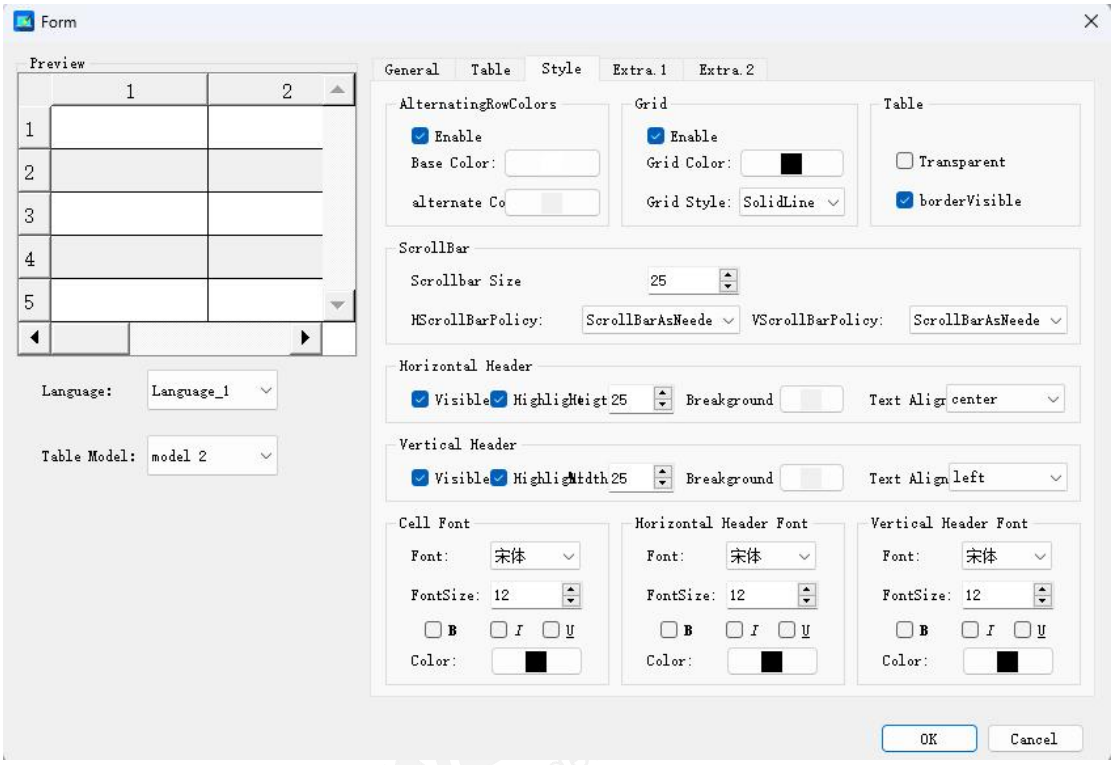
3) Column Width: Set the width of individual columns.

4) Read-Only: Control whether the column cells are read-only.

5) Horizontal Alignment: Horizontal alignment of cells (left, center, right).

6) Vertical Alignment: Vertical alignment of cells (top, center, bottom).

7) Invisible Bit: Control whether the column is visible in the table, and can be linked to system bit variables.




**1. Alternating Row Colors**

1) Enable: Enable table row coloring

2) Base Color: Color for odd-numbered rows

3) Alternating Color: Color for even-numbered rows

**2. Grid**

1) Enable: Enable table grid

2) Grid Color: Grid color property

3) Grid Style: Grid style property, options include solid, dashed, and other styles.

**3. Table**

1) Transparent Background: Set the table's background transparency property

2) Outer Border Visibility: Set the visibility property of the table's outer border.

**4. Scrollbars**

1) Slider Size: Controls the height of the horizontal slider and the width of the vertical slider.
2) Horizontal and Vertical Slider Display Mode: Options include always visible, never visible, or as needed.

**5. Horizontal Header**

1) Visible
2) Highlight: Controls whether to highlight the text of the current column's horizontal header.
3) Height: Controls the height of the horizontal header.
4) Background Color
5) Alignment: Alignment of the horizontal header text.

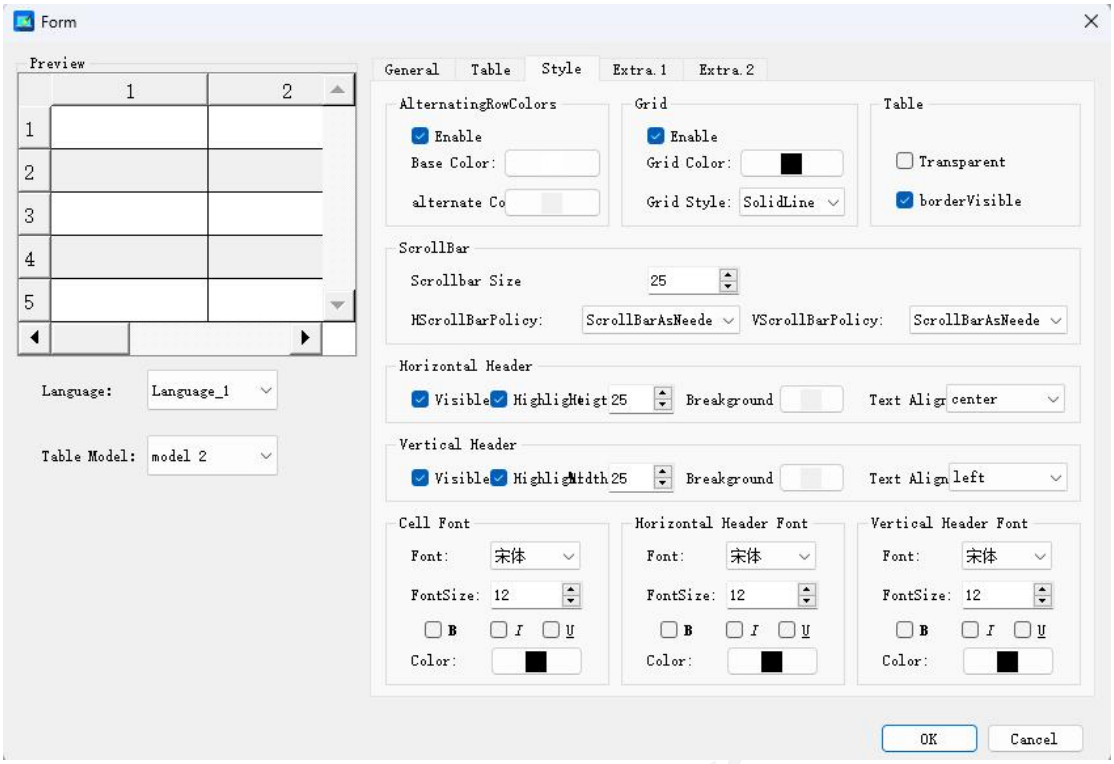
**6. Vertical Header**

1) Visible
2) Highlight: Controls whether to highlight the text of the current row's vertical header.
3) Width: Width of the vertical header.
4) Background Color
5) Alignment: Alignment of the vertical header text.

**7. Cell Font: Uniformly sets the style and color of the table cell fonts.**

**8. Horizontal Header Font: Uniformly sets the style and color of the horizontal header fonts.**

**9. Vertical Header Font: Uniformly sets the style and color of the vertical header**




**1. Basic**

1) Invalid Row Loading: Determines whether to load and display blank rows from the program file in the table.

2) Automatic File Loading: Automatically loads the last opened program file when switching screens.

3) Automatic File Saving: Automatically saves the currently edited program file when leaving the screen.

**2. Row Properties**

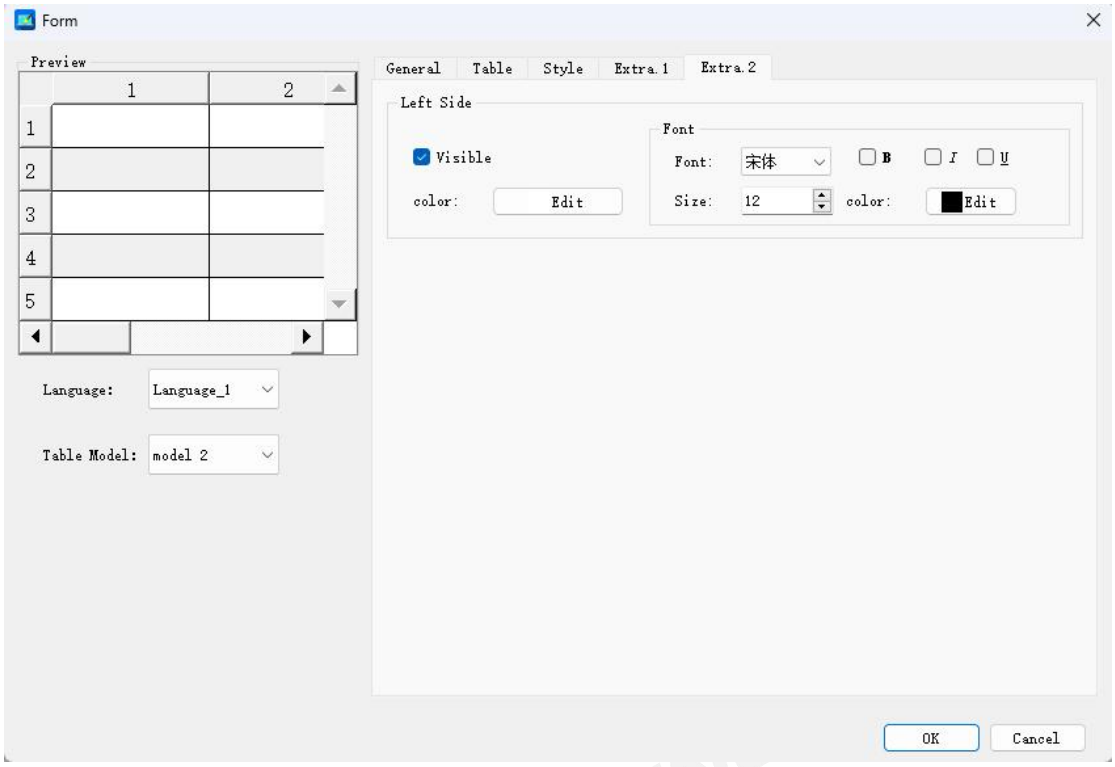
1) Current Row: Switches the current row and allows modification of the corresponding "Fill in Leading Value" property.

2) Fill in Leading Value: Controls whether the leading value corresponding to the "Current Row" is displayed in the table viewport.

3) Alternating Leading Values: Automatically sets the leading values to alternate between rows in the viewport.

**3. Column Properties**

1) Current Column: Displays the column currently being edited.
2) Leading Value: Sets the leading value (i.e., G-code) for the column. The plugin reads the content from the program file and displays it in the table according to the leading values set for each column.
3) Header Text: Sets the header text for the table. The corresponding multilingual text can be edited by right-clicking with the mouse.
4) Value Limits: Each column has corresponding value limits to control the display and input for that column.




**Left Table Header Text Label**

1) Visible: Sets whether the label is visible.
2) Color: Sets the background color of the label.
3) Font: Sets the font style and foreground color of the label.

## 6.37.4. Property Editor


Selection Behavior: Set the selection mode, options include cell, entire row, and entire column. However, the selection mode determines only the manner of selection; accessing the current cell through the table interface retrieves only the cell where the focus is located.

## 6.37.5. Multi-Language Support

The plugin does not support multiple languages.

## 6.37.6. Detailed Description of Functions

The plugin reads the program file and presents the values of corresponding G-codes in a tabular format, showing each line of the program file.

## 6.37.7. Property Interface

| Attribute | Description | Usage |
|---|---|---|
| visible | Read and write table visibility | hTableWidget.visible=false |
| enable | Read and write table enabling | hTableWidget.enable=false |
| geometry | Read and write table geometric dimensions, requires HProperty to read and write specific values | var geometry= hTableWidget.geometry |
| alternatingRowColors | Read and write alternating row colors | hTableWidget.alternatingRowColors=true |
| authority | Read and write table set permissions | hTableWidget.authority=1 |
| columnCount | Get the number of columns in the | var column = |

| | | |
|---|---|---|
| | table set, property read-only | hTableWidget.columnCount |
| gridColor | Read and write table set grid color, requires HProperty to read and write specific values | var color =hTableWidget.gridColor |
| gridStyle | Read and write table set grid style, value corresponds to the index in the property editor dropdown | hTableWidget.gridStyle=0 |
| gridVisible | Read and write table set grid visibility | hTableWidget.gridVisible=true |
| horizontalHeader BackgroundColor | Read and write table set horizontal header background color, requires HProperty to read and write specific values | var color = hTableWidget. horizontalHeaderBackgroundColor |
| horizontalHeaderFont | Read and write table set horizontal header font, requires HProperty to read and write specific values | var font = hTableWidget .horizontalHeaderFont |
| horizontal HeaderFontColor | Read and write table set horizontal header text color, requires HProperty to read and write specific values | var color = hTableWidget. horizontalHeaderFontColor |
| horizontalHeaderHeight | Read and write table set horizontal header height | hTableWidget.horizontalHeaderHeight =35 |
| horizontalHeader HighlightSections | Read and write table set horizontal header highlight | hTableWidget. horizontalHeaderHighlightSections =true |
| horizontalHea | Read and write table set horizontal | hTableWidget.horizontalHeaderVisi |

| | | |
|---|---|---|
| derVisible | header visibility | ble=true |
| horizontalScrollBarPolicy | Read and write table set horizontal scrollbar strategy, value corresponds to the index in the property editor dropdown | hTableWidget.horizontalScrollBarPolicy=0 |
| rowCount | Get the number of rows in the table set, property read-only | var row = hTableWidget.rowCount |
| scrollBarExtent | Read and write table set scrollbar dimensions | hTableWidget.scrollBarExtent=25 |
| softKeypadGeometry | Read and write table set soft keyboard size, requires HProperty to read and write specific values | var geometry= hTableWidget. softKeypadGeometry |
| softKeypadPosition | Read and write table set soft keyboard position, requires HProperty to read and write specific values | var pos= hTableWidget. softKeypadPosition |
| softKeypadType | Read and write table set soft keyboard type, value corresponds to the property editor dropdown value | hTableWidget.softKeypadType=1 |
| stretchLastSection | Read and write table set whether the last column fills the table | hTableWidget.stretchLastSection=true |
| tableWidgetDoubleRowColors | Read and write table set even row color | var color = hTableWidget. tableWidgetDoubleRowColors |
| tableWidgetFont | Read and write table set font, requires HProperty to read and write specific values | var font = hTableWidget.tableWidgetFont |

| | | |
|---|---|---|
| tableWidgetFontColor | Read and write table set text color, requires HProperty to read and write specific values | var color = hTableWidget .tableWidgetFontColor |
| tableWidget SingleRowColors | Read and write table set odd row color, requires HProperty to read and write specific values | var color = hTableWidget. tableWidgetSingleRowColors |
| verticalHeader BackgroundColor | Read and write table set vertical header background color, requires HProperty to read and write specific values | var color = hTableWidget. verticalHeaderBackgroundColor |
| verticalHeaderFont | Read and write table set vertical header font, requires HProperty to read and write specific values | var font = hTableWidget.verticalHeaderFont |
| verticalHeaderFontColor | Read and write table set vertical header text color | var color = hTableWidget. verticalHeaderFontColor |
| verticalHeader HighlightSections | Read and write table set vertical header highlight | hTableWidget. verticalHeaderHighlightSections=true |
| verticalHeaderVisible | Read and write table set vertical header visibility | hTableWidget.verticalHeaderVisible =true |
| verticalHeaderWidth | Read and write table set vertical header width | hTableWidget.verticalHeaderWidth =35 |
| verticalScroll BarPolicy | Read and write table set vertical scrollbar strategy | hTableWidget.verticalScrollBarPolicy=0 |

## 6.37.8. Macro functions

| Prototype | Description | Parameters | Return Value |
|---|---|---|---|

| | | | |
|---|---|---|---|
| cellBackground Color(row,col) | Get the background color of a cell. | (var)row，row (var)col，column | (string) Hex color value. |
| cellChangedCol umn() | Get the column of the cell where content has changed. | None | (int) Column of the cell where content has changed. |
| cellChangedRo w() | Get the row of the cell where content has changed. | None | (int) Row of the cell where content has changed. |
| cellData(row,col ) | Get the value of a specific cell. | (var)row，row (var)col，column | (var) data, value of the cell. |
| cellEnable(row, col) | Get whether the cell is enabled. | (var)row，row (var)col，column | (bool) ok, whether the cell is enabled. |
| cellForeground Color(row,col) | Get the foreground color of a cell. | (int)row，row (int)col，column | (string) color, color name or hex color value. |
| cellsRange() | Get all selected cells. | None | (Array) Cell object values, object members include row and col. |
| clearCellData(ro w,col) | Clear the value of a specific cell. | (var)row，row (var)col，column | None |
| clearContent() | Clear all table content. | None | None |
| clearRowData(r ow) | Clear the value of a specific row. | (var)row，row | None |

| | | | |
|---|---|---|---|
| copyCellData(row,col) | Copy the value of a specific cell. | (var)row，row<br>(var)col，column | None |
| copyRowData() | Copy the value of a specific row. | (var)row，row | None |
| cornerText() | Get the table corner text (available only in Mode Four). | None | (var) cornerText. |
| currentColumn() | Get the current column of the table. | None | (var) current column. |
| currentRow() | Get the current row of the table. | None | (var) current row. |
| cursorDown（） | Next item | None | None |
| cursorLeft（） | Previous item | None | None |
| cursorRight（） | Right item | None | None |
| cursorUp（） | Left item | None | None |
| dataBeforeChange | Get the value of the cell before it was changed. | None | (var) data |
| exportContent (type,path) | Export IoT display content to an Excel file (valid only in Mode Five). | (int)type, export parameter.<br>0 - Export data<br>1 - Export graphics<br>2 - Export data and graphics | None |

| | | | |
|---|---|---|---|
| | | (string)path, export file path. | |
| find(type,content,col, wholeWords, caseSensitive) | Search data within the table (currently available in Mode Four). | (int)type, search type.<br>0 - Text<br>1 - Variable<br>2 - Bit<br>(string)content, search content ("Target"/"USR100"/"COM75962.10").<br>(int)col, specify the search column, default is -1 to search all columns.<br>(bool)wholeWords, whole word match search, default is true.<br>(bool)caseSensitive, case | (Map) Data structure containing cell row and column. |

| | | | |
|---|---|---|---|
| | | sensitive, default is true. | |
| freezeFirstColumn(freeze) | Set whether to freeze the first column of the table. | (bool)freeze, whether to freeze. | None |
| freezeFirstRow(freeze) | Set whether to freeze the first row of the table. | (bool)freeze, whether to freeze. | None |
| getColumnWidth(column) | Get column width. | (int)column. | (int) width |
| horizontalHeaderText (index) | Get the horizontal header text of a specific column. | (int)index, specify a column's horizontal header. | (string) String type horizontal header text. |
| insertRow(row) | Insert a row at a specified position. | (int)row，row | None |
| isCellEmpty(row,col) | Check if the value of a specific cell is empty. | (var)row，row (var)col，column | (bool) value, whether the cell is empty. |
| isColumnHidden(col) | Get whether a specific column is hidden. | (int)col ，column | (bool) hide, bool value. |
| maximum(row,col) | Get the maximum value limit of a cell. | (int) row row (int) col column | (var) maximum. |
| minimum(row,col) | Get the minimum value limit of a cell. | (int) row row (int) col column | (var) minimum. |

| | | | |
|---|---|---|---|
| openFile(path) | Open a specified file path and load it. | (string)path, string type file path. | (bool) type, whether the open was successful. |
| pageDown() | Next page | None | None |
| pageUp() | Previous page | None | None |
| pasteCellData(row,col) | Paste the value of a specific cell; use in conjunction with copy-paste functions. | (int) row row<br>(int) col column | None |
| pasteRowData(row) | Paste the value of a specific row; use in conjunction with copy-paste functions. | (var)row，row | None |
| redo() | Restore the table to its state before the last operation (currently available in Mode Two). | None | None |
| removeRow(row) | Delete a specified row. | (int)row，row | None |
| saveAs(path) | Save table data to a file. | (string)path, string type file path. | None |
| saveFile() | Save table data to a file. | None | None |
| scrollBarValue() | Get the current value of the scrollbar. | None | (int) value, scrollbar value. |
| setCellBackgroundColor (row,col,color) | Modify the background color of a cell. | (int) row row<br>(int) col column<br>(string)color | None |
| setCellData(row | Modify the value of a | (var)row，row | None |

| | | | |
|---|---|---|---|
| ,col,data) | specific cell. | (var)col，column<br>(var)data, value to write. | |
| setCellEnable(row,col,ok) | Set whether a cell is enabled. | (var)row，row<br>(var)col，column<br>(var)ok, whether enabled. | None |
| setCellFont(row, column, fontName,fontSize, bold,italic,underline) | Modify the font of a cell. | (int)row, specify row.<br>(int)column, specify column.<br>(string)fontName, font name.<br>(int)fontSize, font size, default is 12.<br>(int)bold, bold, default is true.<br>(int)italic, italic, default is true.<br>(int)underline, underline, | None |

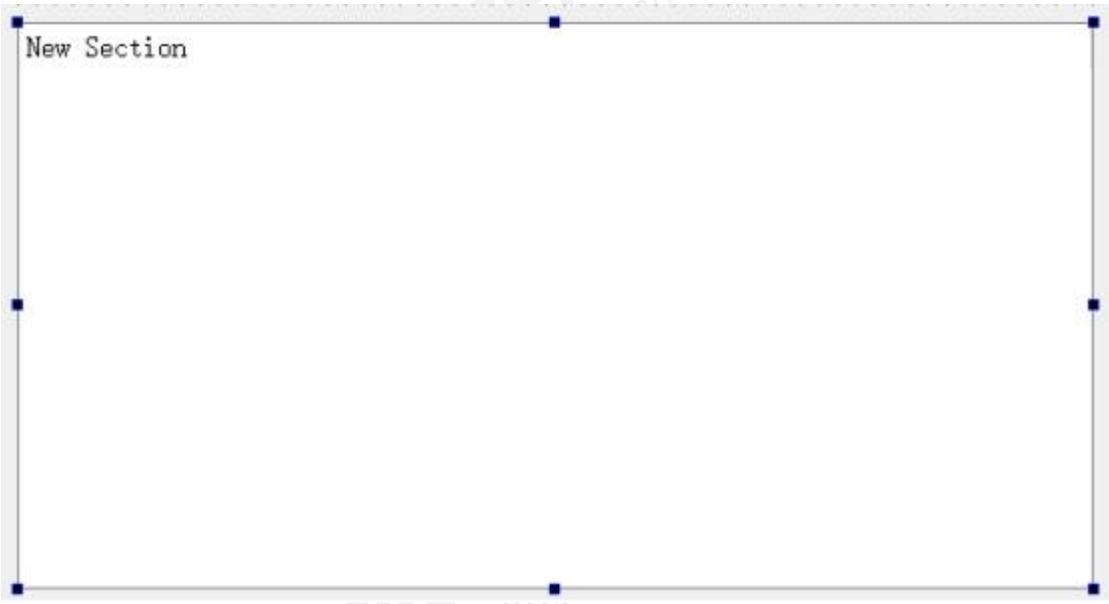
| | | | |
|---|---|---|---|
| | | default is true. | |
| setCellForegroundColor (row,col,color) | Modify the foreground color of a cell. | (int)row，row (int)col，column (string)color | None |
| setColumnHidden(col,true) | Set whether a specific column is hidden. | (int)col ，column (bool)hide，boolvalue | None |
| setColumnReadOnly (col,true) | Set whether a specific column is read-only. | (int)col ，column (bool)true，boolvalue | None |
| setColumnWidth (column,width) | Set column width. | (int)column, specify column. (int)width, column width. | None |
| setCurrentCell(row,col) | Set the current cell of the table. | (int)row，row (int)col,column | None |
| setCurrentColumn(col) | Set the current column of the table. | (int)col,column | None |
| setCurrentRow(row) | Set the current row of the table. | (int)row,row | None |
| setHorizontalHeaderCellFont (index,fontNam | Modify the font of a specific column's horizontal header. | (int)index, specify a column's | None |

| | | | |
|---|---|---|---|
| e,FontSize) | | horizontal header. (string)fontName, font name. (int)FontSize, font size. | |
| setHorizontalHeaderText (index,text) | Set the horizontal header text of a specific column. | (int)index, specify which horizontal header to modify. (string)text, string type header text. | None |
| setMultiSelectionEnable (enable) | Set whether the table allows multiple selections. | (bool)enable, enable multi-selection for table cells. | None |
| setSelectionBehavior(value) | Set the selection behavior of the table. | (int)value: 0: Select cell 1: Select row 2: Select column | None |
| setVerticalHeaderText (index,text) | Set the vertical header text of a specific row. | (int)index, specify which vertical header to | None |

| | | | |
|---|---|---|---|
| | | modify.<br>(string)text,<br>string type<br>header text. | |
| titleText() | Get the table title text. | None | (var) title. |
| totalRowCount() | Get the total number of rows in the table. | None | (int) Total number of rows in the table. |
| Undo() | Undo table operation (currently available in Mode Two). | None | None |

## 6.38. Tree diagram

### 6.38.1. Component Appearance




Components include headers and node content.

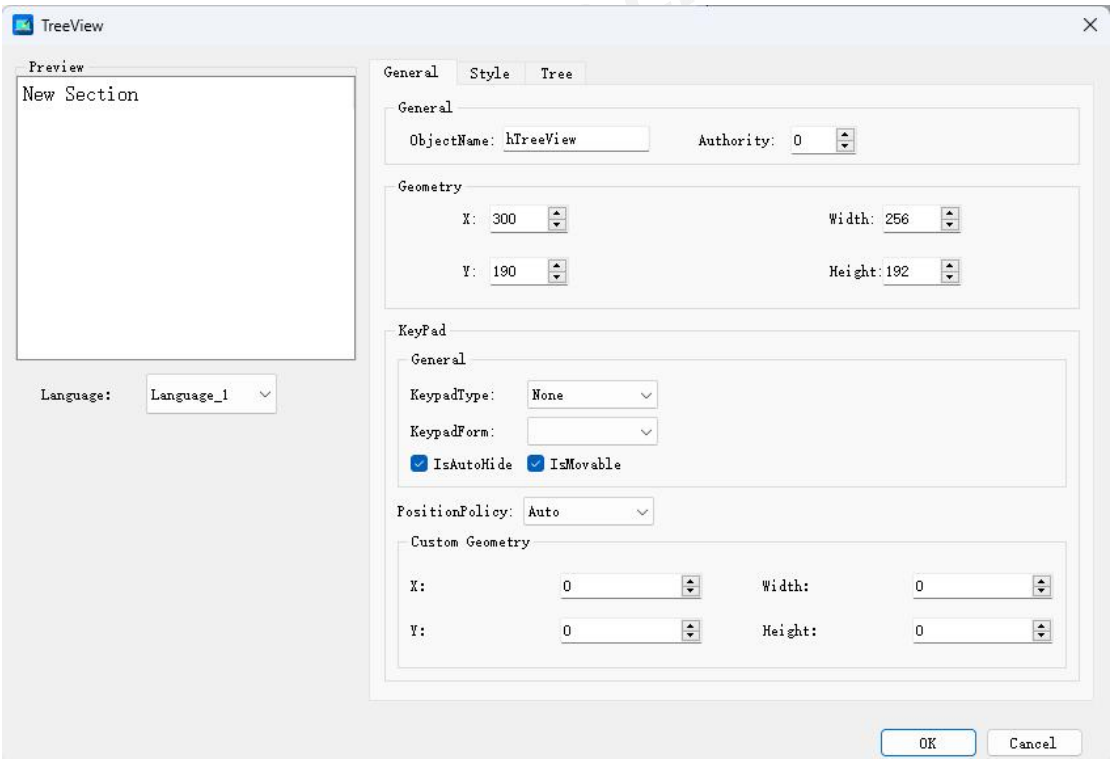
### 6.38.2. Introduction

The plugin presents data in a tree structure.

# 6.38.3. Property Editing Dialog

The property editing dialog includes three tabs: 'Basic', 'Style', and 'Tree'.

[Basics]




**1. Basics**

1) Each object can be identified and accessed by setting an object name. The object name is a string used to reference and manipulate the object in script code.
2) Operation permissions
3) Set operation permissions for the tree diagram plugin.

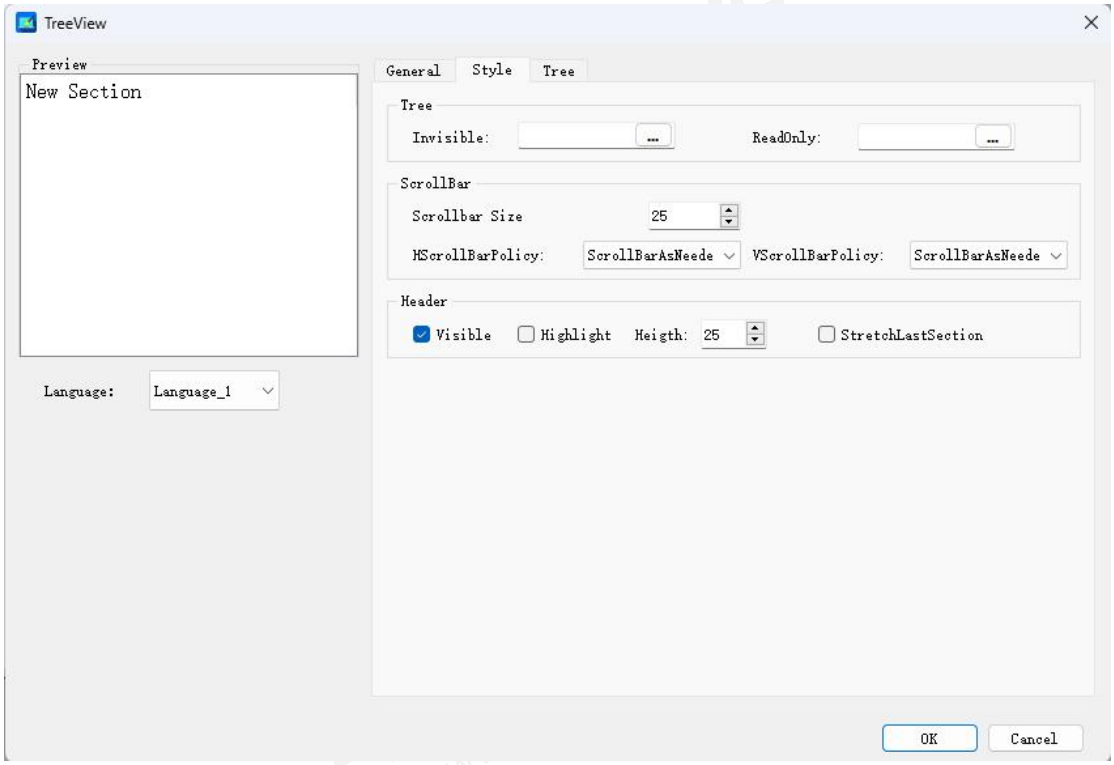
**2. Geometric Dimensions**

Set and save the properties for the position and size of the plugin.

**3. Soft Keyboard**

When using touch input, the soft keyboard can be used (refer to the soft keyboard documentation). By default, the character input keyboard is used.

[Appearance]




### 1. Tree

**Invisible Bit**

The invisible bit is associated with a system address. When the invisible bit is true, the tree diagram is hidden and not visible.

### 2. Read-Only Bit

The read-only bit is associated with a system address. When the read-only bit is true, the tree diagram cannot be edited through the soft keyboard or input blocks.

### 3. Scrollbars

**Scrollbar Size**

Set the size of the vertical and horizontal scrollbars.

**Horizontal Scrollbar Display Mode**

Set the display mode of the horizontal scrollbar with three strategies (show scrollbar when needed, always show scrollbar, never show scrollbar).

**Vertical Scrollbar Display Mode**

Set the display mode of the vertical scrollbar with three strategies (show scrollbar when needed, always show scrollbar, never show scrollbar).

## 4. Header

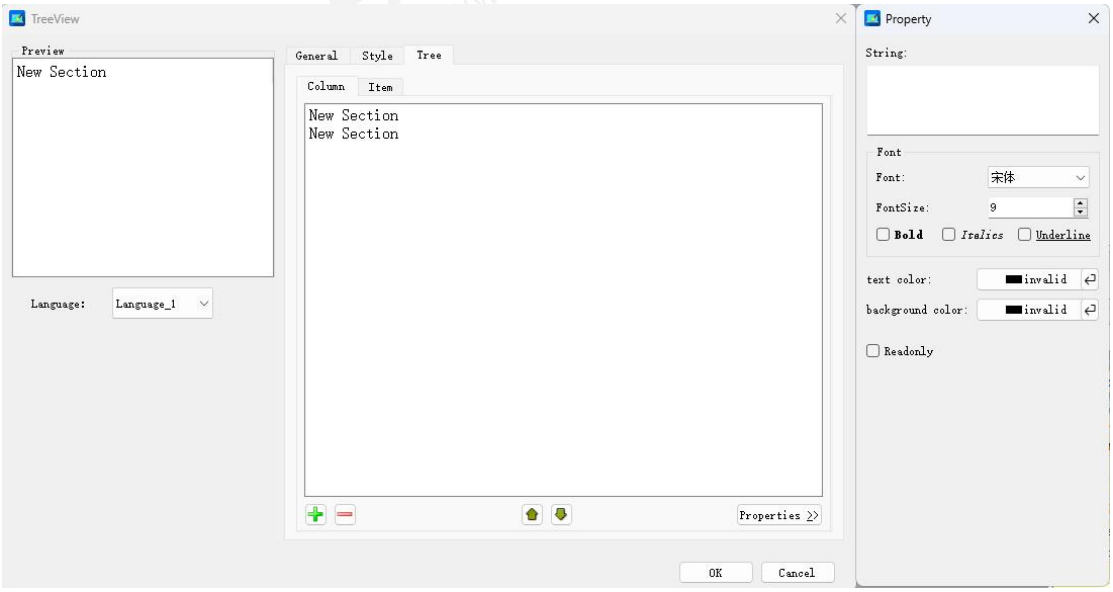
**Visible:** Whether the horizontal header is visible.

**Highlight:** Whether the horizontal header is highlighted.

**Height:** The height of the horizontal header.

Last Column Fill: Whether the last column of the horizontal header fills the viewport.

[Tree]




## 1. Header

In a column table format, set the data for the horizontal header, corresponding to the

functional buttons for adding, deleting, moving up, and moving down the header. Properties include display text, font, text color, and background color.

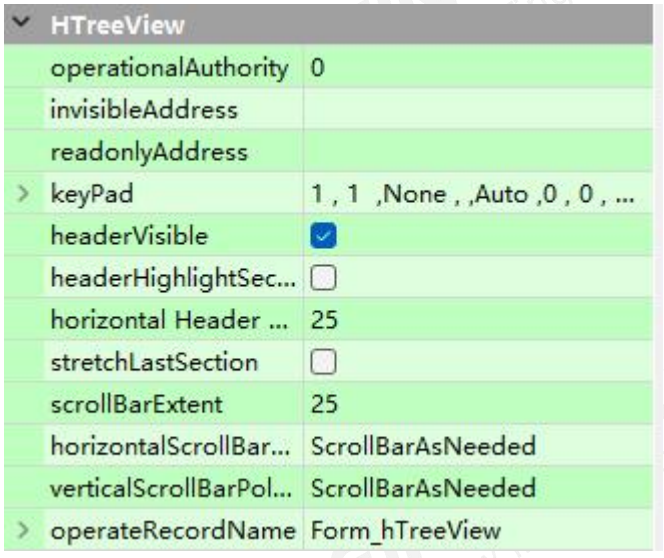
**2. Nodes**

The node page sets the display content for the tree diagram's nodes, with functional buttons for adding, adding child nodes, deleting, moving nodes up one level, moving nodes down one level, and moving nodes up or down. Each node has corresponding properties, including display text, font, text color, and background color.

**3. Language**

The tree diagram supports multiple languages. Switch the language in the preview window to edit the multi-language properties corresponding to the selected preview language.

## 6.38.4. Property Editor




I don't have the capability to display or interact with pop-up windows or specific interfaces directly. If you have specific attributes or properties you'd like to discuss or need help with, feel free to describe them here!

## 6.38.5. System Multilingual Support

The plugin does not support multiple languages.

## 6.38.6. Detailed Function Description

It seems like you're describing a plugin that presents data in a tree-like structure, and it provides interfaces for reading and writing node content such as node text, font, and color settings.

## 6.38.7. Properties

| Properties | Descriptions | Usages |
|---|---|---|
| visible | Read and write plugin visibility attribute. | var visible = Form.hTreeView.visible |
| enable | Read and write plugin enable attribute. | Var enable = Form.hTreeView.enable |
| geomerty | Read and write plugin geometric dimensions attribute, requires HProperty to read and write specific value. | var geomerty = Form.hTreeView.geomerty |
| columnCount | Property read-only, read the number of columns in the tree diagram. | var columnCount = Form.hTreeView.columnCount |

| | | |
|---|---|---|
| headerHeight | Read and write tree diagram header height. | var height=Form.hTreeView.headerHeight |
| headerHighlightSections | Read and write whether the plugin header is highlighted. | var value = hTreeView.headerHighlightSections |
| headerVisible | Read and write whether the plugin header is displayed. | var value = hTreeView.headerVisible |
| operateRecordName | Read and write plugin operation record name. | var value = hTreeView.operateRecordName |
| operationalAuthority | Read and write plugin permissions. | var value = hTreeView.operationalAuthority |
| scrollBarExtent | Read and write scrollbar size. | hTreeView.scrollBarExtent=50 |
| stretchLastSection | Read and write whether the plugin's last column fills the viewport. | hTreeView.stretchLastSection=true |

## 6.38.8. Function Interface

| Function Prototype | Description | Parameters | Return Value |
|---|---|---|---|
| clearSection(parent) | Clear all child nodes under the | • (Index)parent: If not set, clear | None |

| | | | |
|---|---|---|---|
| | parent node. | the tree diagram. Index type variable can be obtained through the function index/currentIndex. | |
| collapesSection(index) | Collapse the target node. | • (Index)index: Target node, can be obtained through the function index/currentIndex | None |
| Collapse() | Collapse all nodes in the tree diagram. | None | None |
| currentIndex() | Get the current node in the tree diagram. | None | (Index) Current node |
| expand() | Expand all nodes in the tree diagram. | None | None |
| expandSection | Expand the node at the specified index in the tree diagram. | (Index)index: Target node, can be obtained through the function index/currentIndex. | None |
| headerText() | Get the header text | (int)column | (string) |

| | | | |
|---|---|---|---|
| | of column column. | | Header text (Index) Target node (int) Row position of the target node based on the parent node |
| Index(row,column,parent) | Get the index of row row column column under the parent node. | (int)row，row (int)column,column (Index)parent: Parent node of the target node. If not set, the root node is used. Index type variable can be obtained through the function index/currentIndex. | (string) Header text (Index) Target node (int) Row position of the target node based on the parent node |
| indexOfParent(parent) | Return the row position based on the parent node. | (Index)target node | (string) Header text (Index) Target node (int) Row position of the target node based on the |

| | | | |
|---|---|---|---|
| | | | parent node |
| insertColumn(column) | Insert a column at column in the tree diagram. | (int)column: Insert a column at column column. | None |
| insertRow(row,parent) | Insert row row under the parent node. | (int)row: Insert at row row.<br>(Index)parent: Insert under parent. If not set, insert under the root node. Index type variable can be obtained through the function index/currentIndex. | None |
| parentSection(index) | Get the parent node of the index node. | (Index)target node | (Index) Parent node of the target node |
| removeColumn(column) | Delete the column column in the tree diagram. | (int)column: Delete column. | None |
| removeRow?2(row ， parent) | Delete row row under the parent node. | (int)row: Delete row row.<br>(Index)parent: Insert under parent. If not set, delete under the root node. Index type variable can be | None |

| | | | |
|---|---|---|---|
| | | obtained through the function index/currentIndex. | |
| sectionText(index) | Get the text of the target node. | (Index)index: Target node, can be obtained through the function index/currentIndex. | (string) Node text |
| setHeaderText(column,text) | Set the header text of column column. | (int)column，column (string) text | None |
| setSectionBackgroundColor(index,color) | Set the background color of the node. | (Index)index: Target node, can be obtained through the function index/currentIndex. (var)color: Color, can be obtained through HProperty. | None |
| setSectionFont(index,font) | Set the font of the target node. | (Index)index: Target node, can be obtained through the function index/currentIndex. (var)font: Font, can be obtained through HProperty. | None |
| setSectionText(index,text) | Set the text of the target node. | (Index)index: Target node, can be obtained through the function index/currentIndex. (string)text: Write text. | None |

| | | | |
|---|---|---|---|
| setSectionTextColor(index,color) | Set the text color of the node. | (Index)index: Target node, can be obtained through the function index/currentIndex.<br>(var)color: Color, can be obtained through HProperty. | None |

## 6.38.9. Signal

| Prototype | Description | |
|---|---|---|
| sectionDataChanged(Index) | Node data changed | Signal Associated Script Function<br>Access the first parameter Index through arguments[0] in the function to get the index of the node where data has changed. |
| sectionClicked(index) | Node clicked | Signal Associated Script Function<br>Access the first parameter Index through arguments[0] in the function to get the index of the node that was clicked. |
| currentSectionChanged(index,index) | Current node changed | Signal Associated Script Function<br>Access the first parameter Index through arguments[0] in the function to get the current node, and access the second parameter Index through arguments[1] to get the previous node. |

## 6.39. List Box

## 6.40. Date Cloc

## 6.41. Dynamic Image

## 6.42. Dynamic Text

## 6.43. Coordinate Display

## 6.44. Historical Alarm Display

## 6.45. Current execution display

## 6.46. Operation Log

### 6.46.1. Feature Introduction

Operation logging" records the usage history of computer systems or applications, including operations performed by users and their corresponding outcomes. It is crucial for diagnosing system failures or anomalies, allowing for the identification of root causes and the implementation of necessary solutions. Additionally, operation logging helps businesses understand user behaviors and demands, analyze preferences and behavioral patterns, and optimize product design and operational strategies accordingly. Through the operation logging component, users can trace back their activity history within systems and applications.

Currently, the following components support user operation logging: table set, code editor, file selector, file transfer, numeric input, integer input, decimal input, button, and dropdown menu.

Operation records provide multiple search methods:

- Search by record creator
- Search by operation content
- Search by details
- Support specifying a search time range

Custom operation records can be added

// Add custom record

// operate: operation

// detail: detail

bool addOperateRecord(operate, detail);

Add an interface in the macro wizard to allow users to add custom operation records in macro programs.

## 6.46.2. Component Structure


The component consists of horizontal headers, vertical headers, table content, horizontal scroll bar, and vertical scroll bar.

## 6.46.3. Property Editor

The operation record component does not provide a property setting dialog, but relevant properties can be set using the property editor.

1. **Scroll bar width or height**
2. **Virtual keyboard**

When using touch or mouse input, a virtual keyboard can be used. (Link to virtual keyboard instructions)

3. **Device**

Specify the device to connect to when using remote visualization.

## 6.46.4. Script Macro

You can find details in the "HOperationRecord_A20_Designer_CN.xml" file located in the macroWizard folder.

## 6.47. Progress Bar

## 6.48. HVisionView (Vision Plugin)

## 6.49. Label

## 6.50. Indicator Light

## 6.51. Frame

## 6.52. Grid Splitter

## 6.53. Option Group

## 6.54. Tab Control

## 6.55. HCam2D

## 6.56. Five-Sided Drill

## 6.57. CAM Plugin

## 6.58. Cam Grinding

## 6.59. SpringEditorPlugins (3D Spring)

**Note: System variables related to enabling functionalities**

To use the 3D spring functionality, ensure that the following system parameters on the controller are enabled

COM[40782] = 1 controls whether the 3D spring feature is enabled.

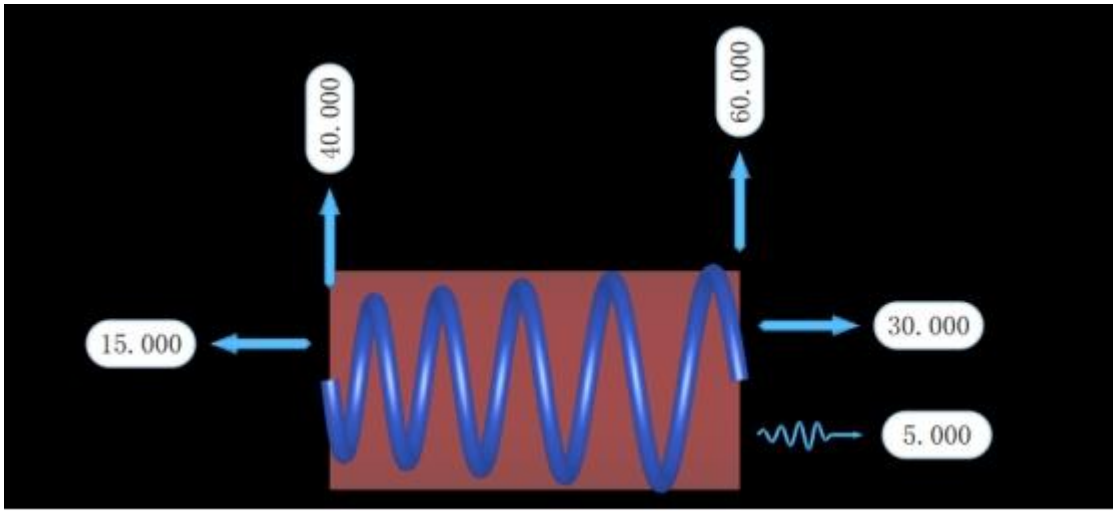
Note: After setting 73831 = 64, you need to click the "Reset" button on Trans and then restart to save this variable. If the value of this variable remains 64 after the restart, it indicates successful activation.


| Parameters | Definition | Range | Unit | Default value | Activation method |
|---|---|---|---|---|---|
| #C73831.6 | Is the 3D spring machine functionality enabled? | 0~1 | Position | 0 | Restart |
| | = 0, not enabled<br>= 1, enabled. Please contact customer service to enable this feature for normal use. Supported from version B0x_V1_2023-09-06-09-16.axBin and later versions. | | | | |

## 6.46.5. Spring Properties

1. Left Outer Diameter
2. Right Outer Diameter

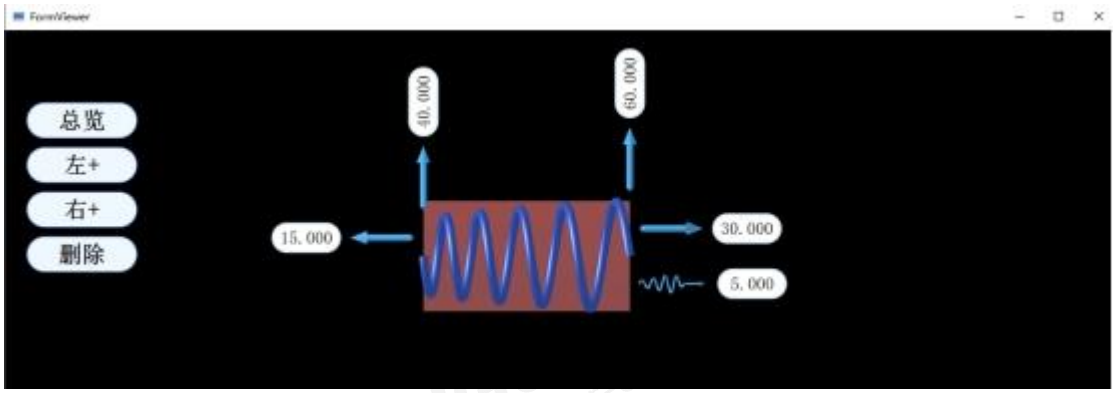
3. Left Pitch
4. Right Pitch
5. Number of Turn




Here is a representation of a section of a spring:Left Outer Diameter: 40;Right Outer Diameter: 60;Left Pitch: 15;Right Pitch: 30;Number of Turns: 5

## 6.46.6. Interactive Operations

### 1. 3D Interface Display




In the 3D interface, only convenient and commonly used simple operations are provided. For more interface functionalities, please use the HMI interface.

### 2. Basic Operations

1. Spring Property Value Modification: Select the spring to edit. Five arrows and the

property value display will appear around the spring segment.

2. Quick Property Modification: Drag the arrows for fast property adjustments, or click on the property value to manually enter the desired value.
3. Delete Spring: After selecting the spring, click the delete button on the left side of the interface to remove the spring. Note: If only one spring remains, it cannot be deleted.
4. Add Spring: After selecting the spring, click the (Left +) button on the left side of the interface to add a segment of spring to the left of the selected spring. The (Right +) button works similarly for adding to the right.
5. Overview: Click the overview button to display the entire spring horizontally in the center of the view.
6. View Operations:

1) Hold the left mouse button and move to rotate the view.
2) Hold the middle mouse button and move to pan the view.
3) Scroll the mouse wheel to zoom in or out of the view.

## 6.46.7. HMI Interface

| Function Name | Description |
|---|---|
| setSpringData( QVariantdataList, double wire_diameter) | Set spring data, including spring segment data in a 2D array and wire diameter |
| setSingleSpringData(int index, QVariant data) | Set data for the spring at index index |
| springData() | Return spring data |
| setBackgroundColor(int r, int g, int b) | Set background color |
| setSpringColor(int index, int r, int g, int b) | Set color for the spring at index index |
| setDefaultTurns(int turns) | Set default number of turns for new springs in 3D editing |
| setRings(int rings) | Set spring arc precision |
| setSlices(int slices) | Set spring cross-section precision |

| | |
|---|---|
| setEditEnabled(bool enabled) | Enable 3D editing |
| setWireDiameter(double diameter) | Set wire diameter |
| wireDiameter() | Return wire diameter |
| springCount() | Return number of spring segments |
| setSelected(index) | Select the spring at index index |
| deleteSpring(index) | Delete the spring at index index |
| addSpring(int index, QVariant data) | Add a spring at index index |
| showAll() | Overview view |
| heightAt(int index) | Return height of the spring at index index |
| totalHeight() | Return total height of the spring |
| lengthAt(int index) | Return wire length of the spring at index index |
| totalLength() | Return total wire length of the spring |
| panAboutViewCenter(double ngle) | Translate the camera around the view center by angle (degrees) |
| tiltAboutViewCenter(double angle) | Tilt the camera around the view center by angle (degrees) |
| rollAboutViewCenter(double angle) | Roll the camera around the view center by angle (degrees) |
| undo() | Undo only affects changes made through the interface for adding, deleting, or modifying spring parameters |
| redo() | Redo only affects changes made through the interface for adding, deleting, or modifying spring parameters |

| Signal | Description |
|---|---|
| editFinished() | Send signal after 3D editing is completed. |

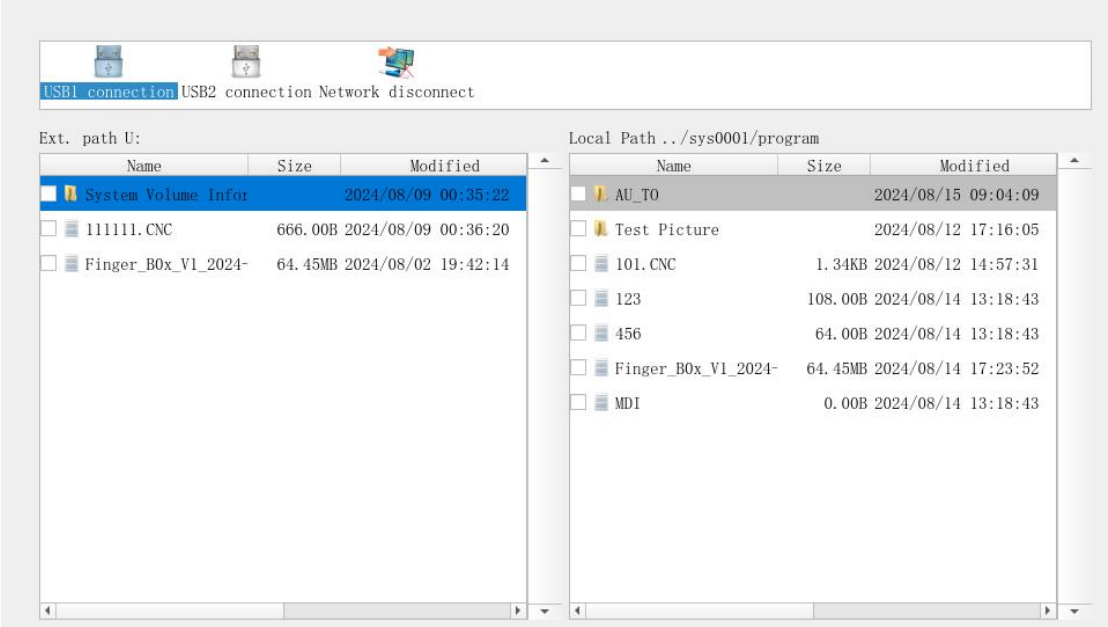
| | |
|---|---|
| indexChanged(int) | Send signal and return the spring index when a spring is selected. |

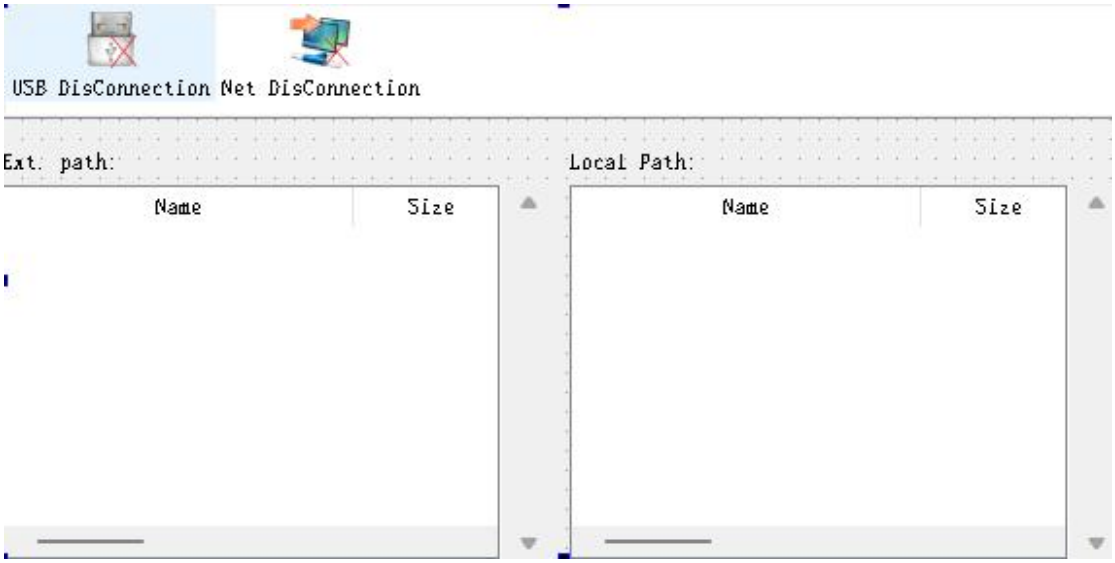
# 6.47. Program Transfer

## 6.47.1. Feature Introduction

The purpose of file transfer is to move a file from one device to another. Common scenarios include transferring files from the controller's local file system to a USB drive, or from a USB drive to the controller. Additionally, file transfer includes common operations such as delete, add, rename, copy, and paste. For more details, refer to "HFilecopy_Designer_CN.xml".

## 6.47.2. Component Structure






**1. Connected Devices Indicator**

Displays the currently connected devices, currently supporting USB devices only; network devices are not supported.

**2. External Path Textbox**

Displays the current directory path of the external device.

**3. Local Path Textbox**

Displays the current directory path of the local device.

**4. External File Information Table**

Displays file information of the external device.

**5. Local File Information Table**

Displays file information of the local device.

## 6.47.3. Property Editor

The file transfer component does not provide a popup for setting properties; you can

configure related attributes using the attribute editor.

### 1. Geometry Size

This property controls the position and size of the component within its parent window.

### 2. Font

This property saves the currently set font for the component.

### 3. Virtual Keyboard

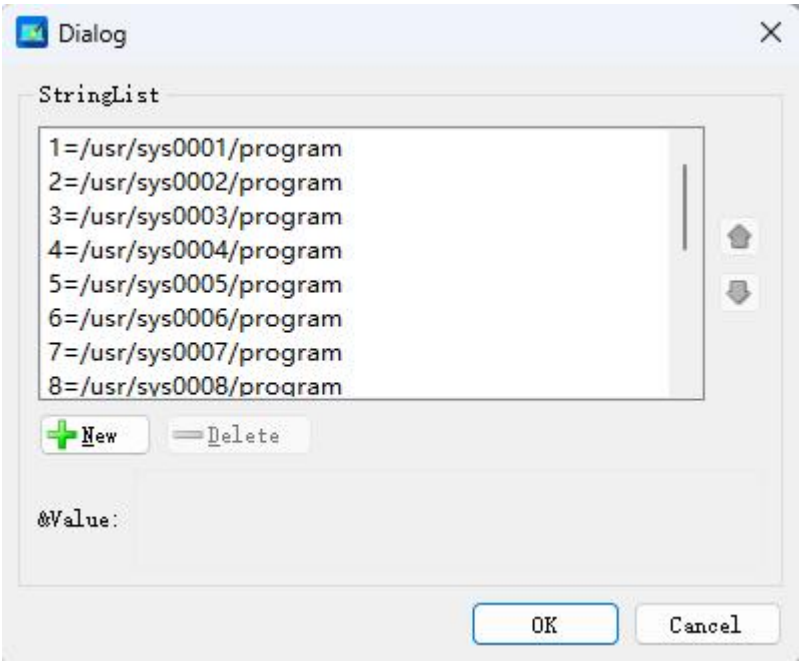
Set the virtual keyboard used by the component.

### 4. Default channel path

This property sets the component to use the restricted path corresponding to the ID number in 'Restricted Path Opening'.

Note: 'Restricted Path Opening' means that only the specified path and its subdirectories can be opened, preventing users from accessing critical system directories and causing accidental operations.

### 5. Restricted path opening




Set restricted path opening.

Note: "Restricted Path Opening" means only allowing access to the specified path and its subdirectories, preventing users from accessing critical system directories and potentially causing accidental operations.

### 6. Cursor color

Sets the color of the cursor in the table.

### 7. Focus Text Color

Sets the text color of the row where the cursor is located.

### 8. Selected Background Color

Sets the background color when a file is selected.

Priority: Cursor Color > Selected Background Color > Alternating Color > Background Color

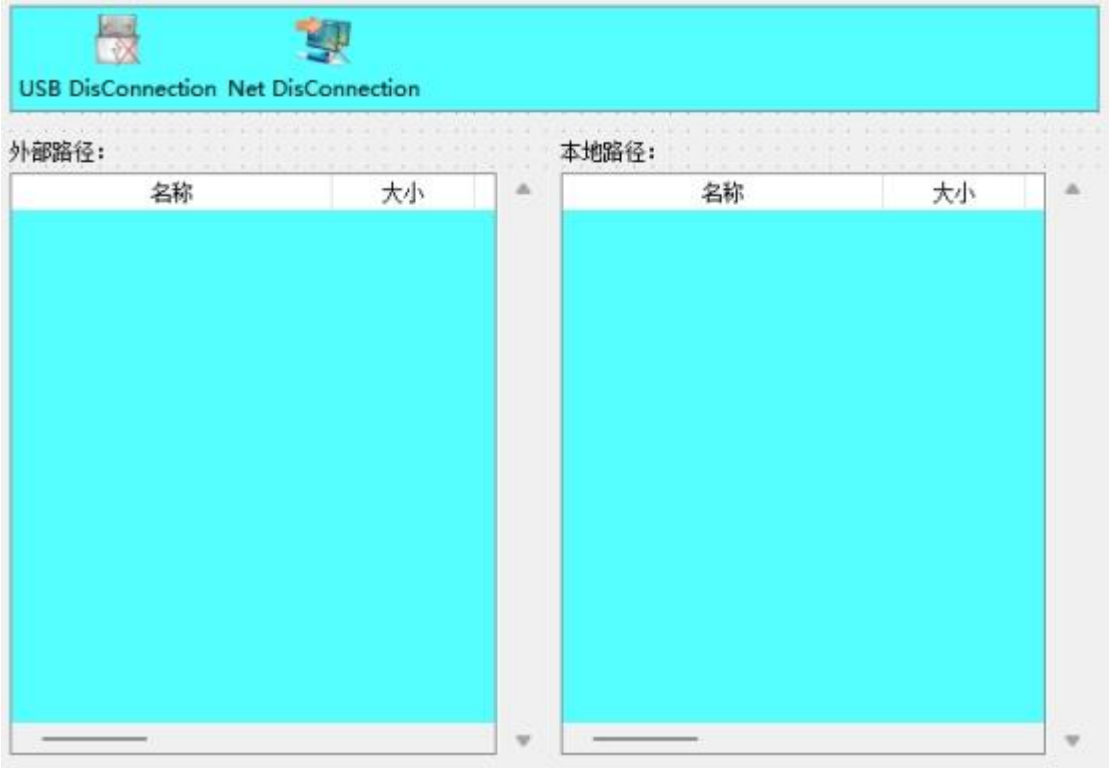
### 9. Text Color

Sets the text color in the table.

Priority: Focus Text Color > Text Color

### 10. Background Color

Sets the background color of the component.




### 11. First Alternating Color

Sets the first alternating color for the table.

Priority: Cursor Color > Selected Background Color > Alternating Color > Background Color

### 12. Second Alternating Color

Sets the second alternating color for the table.

### 13. Horizontal Header Text Color

Sets the text color of the table's horizontal header.

**14. Horizontal Header Background Color**

Sets the background color of the table's horizontal header.

**15. Device**

Sets which device's files are displayed when visualizing remotely.

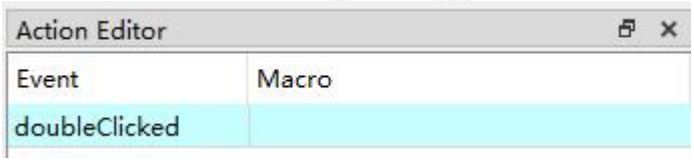
**16. Operation Records**

Sets whether the operation record feature for this component is enabled. See "Operation Records" for details.

**17. Operation Record Name**

Sets the name of the component in the operation records. See "Operation Records" for details.

## 6.47.4. Action Editor




Double-clicking on a row in the table triggers this event.

## 6.47.5. Script Macros

Details can be found in "HFilecopy_Designer_CN.xml".

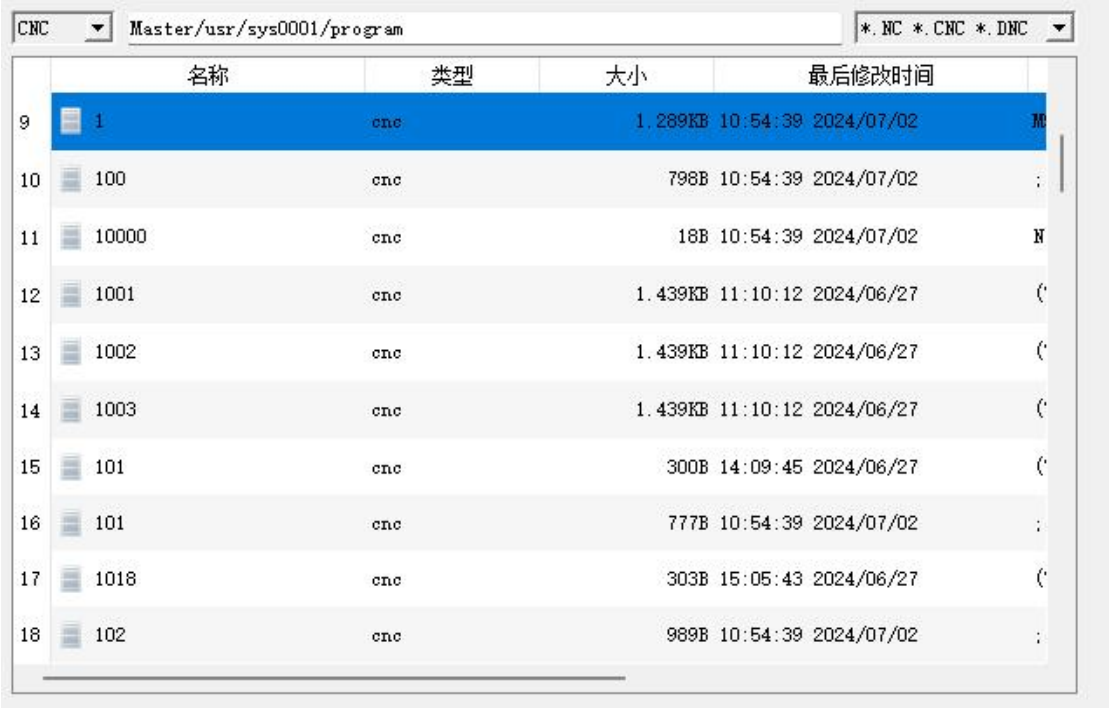
## 6.48. File Selector

## 6.48.1. Feature Introduction

The file selector is used to choose files from the file system. It allows users to browse the

file system and select one or more files. Additionally, the file selector provides common file operations such as delete, add, rename, copy, and paste. For more details, refer to "HFiledialog_Designer_CN.xml".

## 6.48.2. Component Structure




### 1. File Location

When operating in the controller, files can originate from local files, USB drives (after inserting the USB drive), or remote PCs (by mounting PC files to the controller). You can switch using the "File Location" dropdown menu.

### 2. File Path

Displays the current folder path.

### 3. File Type

Displays only files of the specified type in the table.

**4. File Table**

Displays file information.

## 6.48.3. Property Editor

The file selector component does not provide a popup for setting attributes; you can configure related properties using the attribute editor.

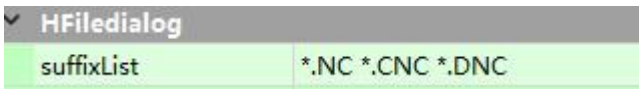
**1. Geometry Size**

This property controls the position and size of the component within the parent window. See “Geometry” for detailed parameters.

**2. Font**

This property stores the font settings for the component. See “Font” for detailed parameters.

**3. File Open Type**




Sets the types of files visible in the file selector. Different types are separated by spaces.

**4. Visibility of “Select” Column**

Sets whether the “Select” column is visible in the table.

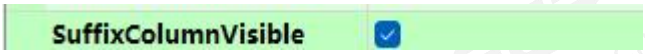
**5. File Name Display**


Sets whether the “Name” column is visible in the table.

### 6. Display file type


Sets whether the “Type” column is visible in the table.

### 7. Display file size




Sets whether the “Size” column is visible in the table.

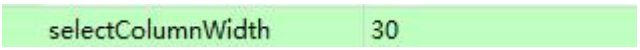
### 8. Last Modified Date Display


Sets whether the “Last Modified Time” column is visible in the table.

### 9. File Information Display


Sets whether the “Info” column is visible in the table.

### 10. “Select” Column Width




Sets the width of the “Select” column in the table.

### 11. File Name Column Width


Sets the width of the “Name” column in the table.

### 12. File Type Column Width


Sets the width of the “Type” column in the table.

### 13. File Size Column Width

| | |
|---|---|
| SizeColumnWidth | 100 |

Sets the width of the “Size” column in the table.

### 14. Last Modified Date Column Width

| | |
|---|---|
| ModifiedColumnWidth | 180 |

Sets the width of the “Last Modified Time” column in the table.

### 15. File Information Column Width

| | |
|---|---|
| InfomationColumnWidth | 200 |

Sets the width of the “Info” column in the table.

### 16. File Location Visibility

| | |
|---|---|
| **fileLocationVisible** | ☑ |

Sets whether the “File Location” dropdown menu is visible.

### 17. File Path Display

| | |
|---|---|
| FilePathVisible | ☑ |

Sets whether the “File Path” text box is visible.

### 18. File Type Filter Display

| | |
|---|---|
| SuffixComboBoxVisible | ☑ |

Sets whether the “File Type” dropdown menu is visible.

### 19. Last Modified Date Format

| | |
|---|---|
| m_fileModify | HMSYMD |

Sets the display format for “Last Modified Date” in the table:

YMDHMS: Year, Month, Day, Hour, Minute, Second

HMSYMD: Hour, Minute, Second, Year, Month, Day

### 20. Soft Keyboard

| keyPad | 1 , 1 ,None , ,Auto ,0... |
|---|---|
| IsAutoHide | ☑ |
| IsMovable | ☑ |
| KeypadType | None |
| KeypadForm | |
| PositionPolicy | Auto |
| X | 0 |
| Y | 0 |
| Width | 0 |
| Height | 0 |

Sets the soft keyboard used by the component. See “Soft Keyboard” for details.

### 21. Default Path for Channels

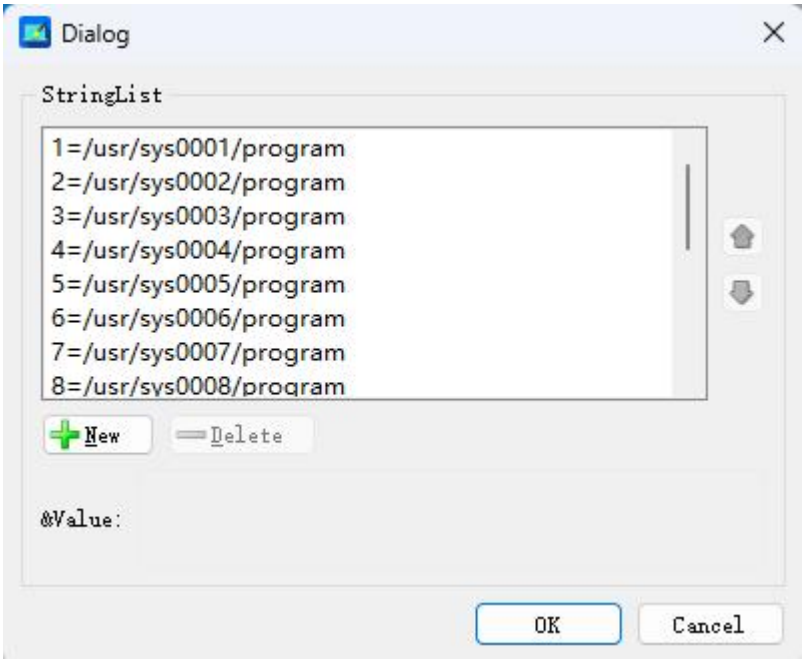
| Default Path | 1 |
|---|---|
| Protect Path | 1=/usr/sys0001/prog... |

Sets the restricted path for the component using the corresponding ID number from “Restricted Open Path”. Note: “Restricted Open Path” limits access to that path and its subdirectories to prevent accidental access to critical system directories.

### 22. Restricted Open Path

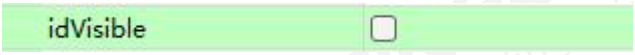
| Default Path | 1 |
|---|---|
| Protect Path | 1=/usr/sys0001/prog... |

Sets the restricted open path.

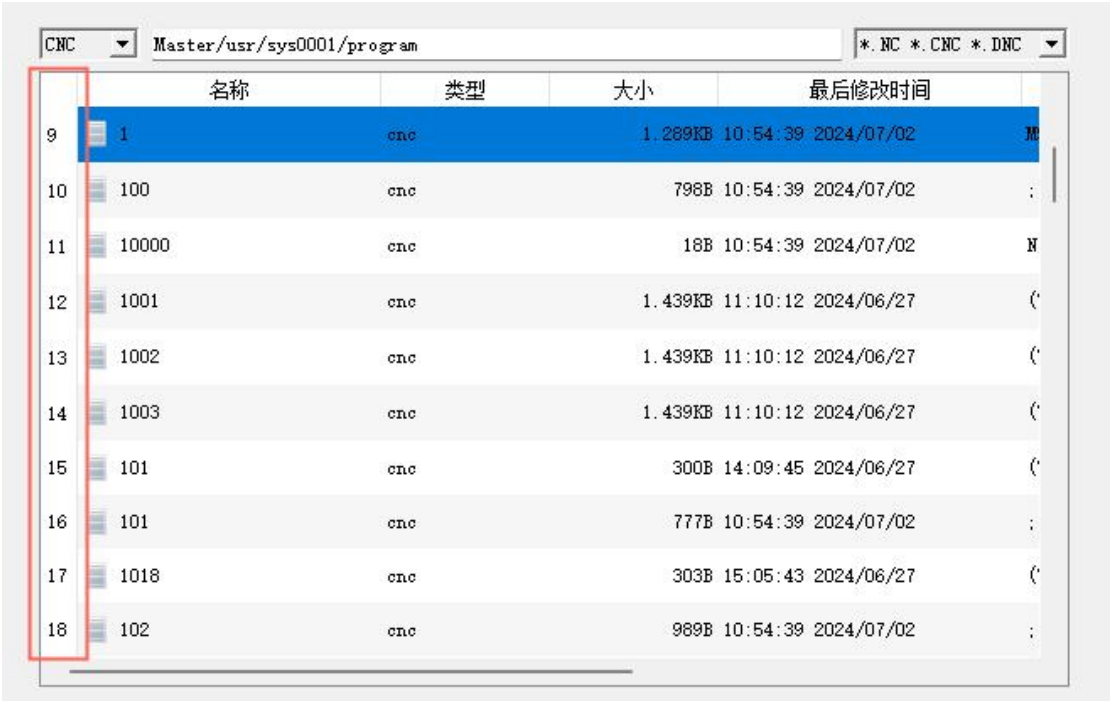



Note: This restricts access to that path and its subdirectories to prevent accidental access to critical system directories.

### 23. File Number Display

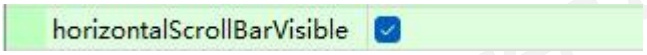



Sets whether file numbers are displayed in the table.



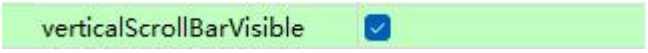
Note: "Restricted Path Opening" means that only the specified path and its subdirectories can be opened, preventing users from accessing critical system directories and causing accidental operations.

### 24. Horizontal Scrollbar Display




Sets whether the horizontal scrollbar is visible in the table.

### 25. Vertical Scrollbar Display

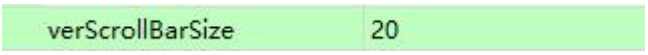



Sets whether the vertical scrollbar is visible in the table.

### 26. Horizontal Scrollbar Size

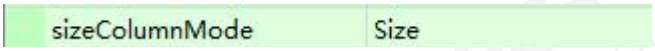

Sets the height of the horizontal scrollbar.

### 27. Vertical Scrollbar Size




Sets the width of the vertical scrollbar.

### 28. File Size Column Mode



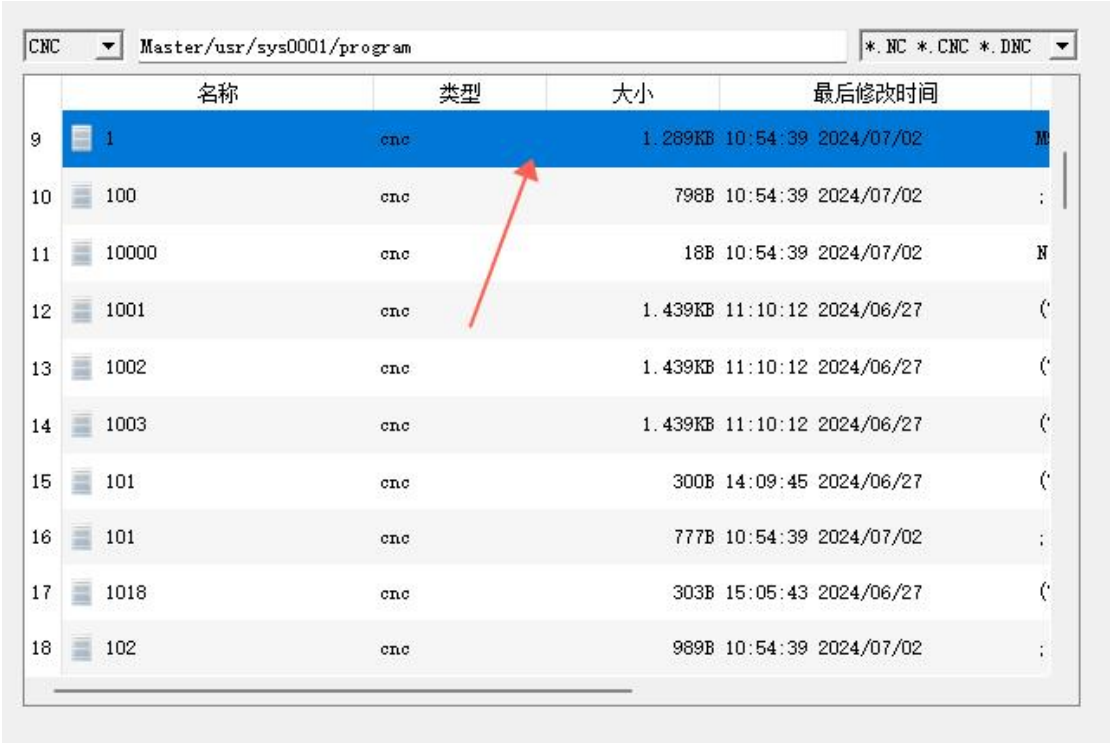

Sets the content displayed in the “Size” column of the table:

Size: Displays file size

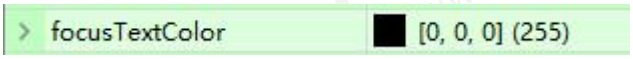
LineCount: Displays the number of rows in the file

### 29. Cursor Color


Sets the color of the cursor in the table.

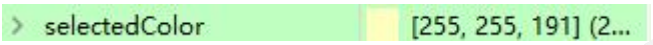



### 30. Focus Text Color




Sets the text color of the row where the cursor is located.

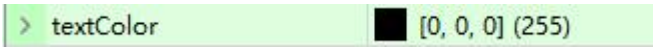
### 31. Selected Background Color




Sets the background color of the selected file.

Priority: Cursor Color > Selected Background Color > Alternate Color

### 32. Text Color




Sets the text color in the table.

Priority: Focus Text Color > Text Color

### 33. First Alternate Color

| | |
|---|---|
| firstAlternateColor | [255, 255, 255] (2... |
| secondAlternateColor | [246, 246, 246] (2... |

Sets the first alternate color for the table.


### 34. Second Alternate Color

| | |
|---|---|
| firstAlternateColor | [255, 255, 255] (2... |
| secondAlternateColor | [246, 246, 246] (2... |

Sets the second alternate color for the table.

### 35. Horizontal Header Text Color

| | |
|---|---|
| horHeadTextColor | [0, 0, 0] (255) |
| horHeadBackGroundColor | [255, 255, 255] (2... |

Sets the text color of the horizontal table header.

### 36. Horizontal Header Background Color


Sets the background color of the horizontal table header.

### 37. Vertical Header Text Color


Sets the text color of the vertical header.

### 38. Vertical Header Background Color


Sets the background color of the vertical header.

### 39. Custom Icon


Sets the icon used for files. Use with the setUserIconEnable interface.

// Set custom icon for files

// filePath: File path

// enable: true - Use custom icon, false - Use default icon

void setUserIconEnable(const QString &filePath, bool enable);

### 40. Device


Sets which device's files are displayed during remote visualization.

### 41. Operation Record

| | |
|---|---|
| operateRecordEnable | ☐ |
| **operateRecordName** | Form_hFiledialog |

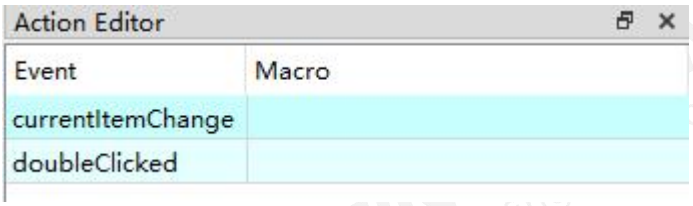
Sets whether the operation record function for the component is enabled. See “Operation Record” for details.

### 42. Operation Record Name

| | |
|---|---|
| operateRecordEnable | ☐ |
| **operateRecordName** | Form_hFiledialog |

Sets the name of the component in the operation record. See “Operation Record” for details.

## 6.48.4. Action editor




### 1. Current selection changed

When the current selection changes in the table, this event is triggered.

### 2. Double-click

Double-clicking a row in the table triggers this event.

## 6.48.5. =Script Macro

Details can be seen.“HFiledialog_Designer_CN.xml”

## 6.49. Parameter monitoring table

## 6.50. Custom script jump

## 6.51. Servo Drive parameters

## 6.52. Keyboard container

## 6.53. Keyboard key

## 6.54. Keyboard display

# 7. Basic macro functions

## 7.1. Macro editor

## 7.2. System Function group

The System Functions Group lists the basic system functions that allow you to perform fundamental operations on the system.

### 7.2.1. Retrieve the value of a variable

The getData function is used to retrieve the value of system variables. Its return value provides the current value of the corresponding system variable.

**Usage**

System.getData(channel,dataClass,address)

**Parameter Description**

- Channel："Numeric type, with a value range of [1, 16]."
- dataClass (system variable type): Numeric type, with a value range of [0, 5].
- The following are the meanings corresponding to each value of dataClass:

=0 User；

=1 Mcm；

=2 Sys；

=3 Reg；

=4 Bus；

=5 Com；

- address (Variable Address): Numerical type

**Example**

Example 1: Retrieving the value of #U100 from Channel 1:

Define a variable to store the value of #U100.

var USR100=System.getData(1,0,100)

Example 2: Retrieving the value of #R200 from Channel Define a variable to store the value of #R200.

var REG200=System.getData(2,3,200)

## 7.2.2. Set the value of a variable

The setData macro is used to set system variables and does not return a value.

**Usage**

System.setData(channel,dataClass,address,value)

**Parameter Description**

- channel (channel number): Numeric type, with a value range of [1, 16].
- dataClass (system variable type): Numeric type, with a value range of [0, 5].

- The meanings corresponding to each value of dataClass are as follows:

=0 User；

=1 Mcm；

=2 Sys；

=3 Reg；

=4 Bus；

=5 Com；

- address (Variable Address): Numeric type
- value (Value to be Written): Floating-point type

**Example**

Example 1: Setting #U100 in Channel 1 to 10: `System.setData(1, 0, 100, 10)`

Example 2: Setting #R100 in Channel 2 to 50: `System.setData(2, 3, 100, 50)`

## 7.2.3. Retrieve the value of a variable bit

To retrieve the status of a specific bit in a system variable, returning a boolean value where true indicates the bit is ON and false indicates the bit is OFF.

**Usage**

System.getBit(channel, bitType, address, bitIndex)

**Parameter Description**

- channel (channel number): Numeric type, with a value range of [1, 16].
- bitType (data bit type): Numeric type, with a value range of [0, 12].
- Table 4.1.3.1-1 lists the meanings corresponding to each value of bitType.

=0 I_BIT；

=1 O_BIT；

=2 C_BIT；

=3 S_BIT；

=4 A_BIT；

=5 Cn_BIT；

=6 Tm_BIT；

=7 User_BIT；

=8 Mcm_BIT；

=9 Sys_BIT；

=10 Reg_BIT；

=11 Bus_BIT；

=12 Com_BIT；

- （Set the data address）：Numeric type
- （Bit index):Numeric type, range [0, 31]. Only needs to be set if the data bit type is User, Mcm, Sys, or Reg, indicating the system variable bit number to be configured.

**Example**

Example

Example 1: Retrieve the status of #ABIT1 in Channel 1

var ABit1= System.getBit(1, 4, 1)

Example 2: Retrieve the status of the second bit of #U100 in Channel 1.

var ABit1= System.getBit(1, 7, 100,2)

## 7.2.4. Set the channel data bit to 0 or 1.

Used to set the state of a specific bit in a designated system variable, with no return value.

**Usage**

System.setBit(channel, bitType, address, bitIndex,bit)

**Parameter Description**

- bit (value to be set for the bit): Numeric type, range [0, 1]. When set to 0, it indicates the bit is OFF; when set to 1, it indicates the bit is ON.

- channel, bitType, address, bitIndex, and other parameters are detailed in section 4.1.3.1

**Example**

Example 1: Set #ABIT1 in Channel 1 to ON.

System.setBit(1,4,1,1)

Example 2: Set the second bit of #U100 in Channel 1 to OFF.

System.setBit(1, 7, 100,2.0)

## 7.2.5. Save data

Save data; no return value. Used to store custom data.

**Usage**

System.saveData(ident,value)

**Parameter Description**

- ident (data identifier): Character type
- value (data to be saved): Can be numeric, string, variable, or component property.

**Example**

Example 1: Save data with identifier "count" and value 100.

System.saveData("count",100)

Example 2: Save the "checked" property of the control named "checkBox1" in the screen "Form".

System.saveData("checked",Form.checkBox1.checked)

## 7.2.6. To read data

To read data that was saved using the saveData macro, and where the returned value is the retrieved data

**Usage**

System.loadData(ident)

**Parameter Description**

- ident (data identifier): Character type

**Example**

Example 1: Reading data with the identifier "count".

Var count = System.loadData(“count”)

## 7.2.7. ColorFromString (Convert String to Color Value)

eConvert a string to a color value, with the return value being the converted color value.

**Usage**

System.colorFromString(name)

**Parameter Description**

- name (color value): Character type, can be in the following formats:

Each R, G, B is represented by a hexadecimal character.

#RRGGBB

#RRRGGGBBB

#RRRRGGGGBBBB

Standard color names can be found at the following website:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

**Example**

Example 1: Convert the string "#00ff00" to a color value.

System.colorFromString("#00ff00")

## 7.2.8. Load an executable program file.

Load and execute program" is used to load CNC programs into the controller and initiate execution. It does not return a value.

**Usage**

System.loadExeProgramFile(channel,filename)

**Parameter Description**

- channel (channel number): Numeric type, range [1, 16]
- filename (file path and name): String, enclose this parameter in double quotes when filling it out.

**Example**

Example 1: Load the program named "T1.CNC" into Channel 1.

System.loadExeProgramFile(channel,"/usr/sys0001/program/T1.CNC")

## 7.2.9. Store the specified variable.

Storing specified variable. This macro returns true if the specified channel number or data type is correct; otherwise, it returns false.

**Usage**

System.saveVar(channel,dataType,filename)

**Parameter Description**

- channel (channel number): Numeric type, range [1, 16]
- dataType (data type): Integer type, with specific meanings as follows

=0 specifies the fixed path for user variables of the specified channel:/usr/sys00xx/var/userCurrent.var；

=1 specifies the fixed path for MCM (Multi-Channel Memory) of the specified channel:./usr/sys00xx/var/mcmCurrent.var

=2 specifies the fixed path for system variables (first 10,000) of the specified channel../usr/sys00xx/var/sysCurrent.var

=3 specifies the fixed path for registers of the specified channel../usr/sys00xx/var/regCurrent.var

=4 specifies the fixed path for a common bus../usr/sys0000/var/comCurrentBus.var

=5 specifies the fixed path for common user variables../usr/sys0000/var/comCurrentUser.var

=6 specifies the fixed path for common MCM (Multi-Channel Memory)../usr/sys0000/var/comCurrentMcm.var

=7 specifies the fixed path for common system variables (first 30,000)../usr/sys0000/var/comCurrentSys.var

=8 specifies the fixed path for common registers. ./usr/sys0000/var/comCurrentReg.var

**Note:**

1. Note that in sys00xx, xx represents the channel number. For instance, channel 1 is represented as sys0001, and channel 12 is represented as sys0012.
2. When the data type is between 4 and 8 inclusive, the channel number can be specified as 0 or any value that is not less than the maximum channel number available.
3. filename (file path and name for storage): String. When this parameter is set to an empty string ("") or left blank, it indicates storage to a fixed path.

**Example**

Example 1: Store user variables of Channel 1 in a fixed path.

System.saveVar(1,0)

Example 2: Store MCM variables of Channel 1 in "./usr/sys0001/val.var".

System.saveVar(1,1," /home/root/hust/usr/sys0001/val.var")

## 7.2.10. LoadVar(Load the variable for the specified channel)

Load specified channel variables. This macro returns true if the specified channel number or data type is correct; otherwise, it returns false.

**Usage**

System.loadVar(channel,datatype, fileName)

**Parameter Description**

Please refer to section 4.1.9.1 for parameter details.

**Example**

Example 1: Load user variables of Channel 1.

System.loadVar(1,0)

Example 2: Load MCM variables stored in "./usr/sys0001/val.var" of Channel 1.

System.loadVar(1,1," /home/root/hust/usr/sys0001/val.var")

## 7.2.11. InitVar (Reset variable to factory default)

To restore system variables to their factory settings, this macro returns true if the specified channel number or data type is correct; otherwise, it returns false.

**Usage**

System.initVar(channel,dataType)

**Parameter Description**

- channel (channel number): Numeric type, range [1, 16]
- dataType (data type): For specific details, please refer to section 4.1.9.1.

**Example**

例 1：Example 1: Restore user variables of Channel 1 to factory settings.

System.initVar(1,0)

## 7.2.12. GetProgramDir (Get the program storage path for the specified channel)

To obtain the program storage path for a specified channel, and receive the program storage path as a string upon successful execution of the macro

**Usage**

programDir = System.getProgramDir(channel)

**Parameter Description**

- channel (Channel number): Character type, with value range from [1,16]

**Example**

Example 1: Retrieve the program storage path for Channel 1.

var programDir

programDir = System.getProgramDir(1)

After executing the row, the program storage path for channel 1 (/usr/sys0001/program) is loaded into the variable programDir.

## 7.2.13. GetExePrgFileMain (Get the currently executing main program file name for the specified channel)

To obtain the filename (including the path) of the main program currently being executed on a specified channel, and have the macro return this filename as a string.

**Usage**

System.getExePrgFileMain(channel)

**Parameter Description**

- channel (channel number): Numeric type, range [1, 16]

**Example**

Example 1: Retrieve the filename (including the path) of the main program currently executing on Channel 1.

var cncName = System.getExePrgFileMain(1)

## 7.2.14. GetExePrgFileSub (Get the currently executing subprogram file name for the specified channel)

This macro is used to retrieve the filename (including the path) of the subroutine currently executing on a specified channel. The return value is the filename (including the path) of the subroutine.

**Usage**

System.getExePrgFileSub(channel)

**Parameter Description**

- channel (channel number): Numeric type, range [1, 16]

**Example**

Example 1: Retrieve the filename (including the path) of the subroutine currently executing on Channel 1.

var cncName = System.getExePrgFileSub(1)

## 7.2.15. GetMDItext (Get MDI content)

To retrieve the MDI (Manual Data Input) content and receive the specified MDI content as a string

**Usage**

System.getMDItext(channel)

**Parameter Description**

- channel (channel number): Numeric type, range [1, 16]

**Example**

Example 1: Retrieve the MDI content for Channel 1 and display it in the code display widget on the "Form" screen.

Form.codeDisplay.setTxt(System.getMDItext(1))

## 7.2.16. ExeMDI (Execute MDI)

To execute MDI (Manual Data Input) commands with no return value

**Usage**

System.exeMDI(channel)

**Parameter Description**

- channel (Channel number): Numeric type, with value range from [1,16]

**Example**

Example 1: Execute MDI on Channel 1.

System.exeMDI(1)

## 7.2.17. SetMDItext (Set MDI content)

To set MDI (Manual Data Input) content with no return value

**Usage**

System.setMDItext(channel,string)

**Parameter Description**

- channel (channel number): Integer type, range [1, 16]
- string (text content): Character type

**Example**

Example 1: Set the text content in Channel 1 to "hello, world".

System.setMDItext(1,” hello,world”)

## 7.2.18. Sleep (Delay function)

A delay function to pause execution for a specified amount of time before continuing with the next macro program, with no return value

**Usage**

System.sleep(time)

**Parameter Description**

- time (delay time): Numeric type, unit in milliseconds (ms)

**Example**

Example 1: Save data A and then save data B 1 second later.

System.saveData(“A”,100)

System.sleep(1000)

System.saveData(“B”,50)

## 7.2.19. Execute (Execute external program)

Used to execute an external program, returning either an exception value or an exit code from the program. A return value of -2 indicates that the program cannot be executed, -1 indicates a program crash, and any other value represents the program's exit code.

**Usage**

System.execute(program)

**Parameter Description**

- program (program path): Character type

**Example**

Example 1: Running the program "example" located under the path "/home"

System.execute("/home/example")

## 7.2.20. ResetCNC (Reset controller for the specified channel)

Resetting controller for specified channel. This function is similar to C1. Avoid frequent invocation using timers. No return value.

**Usage**

System.resetCNC(channel)

**Parameter Description**

- channel (channel number): Decimal integer. Specifies the controller channel number according to the bit parameters: BIT00=1 for Channel 1, BIT01=1 for Channel 2, ..., BIT07=1 for Channel 8.

**Example**

Example 1: Reset channels 1, 2, and 3 of the controller. Set BIT0, BIT1, and BIT2 to 1, which translates to the binary number 0111 and in decimal is represented as 7.

System. resetCNC(7)

Example 2: Reset channel 2 of the controller. Set BIT1 to 1 and all other bits to 0, which corresponds to the binary number 0010 and in decimal is represented as 2.

System. resetCNC(2)

## 7.2.21. GetDatas (Get multiple consecutive variable values)

Retrieve consecutive values of multiple variables. This macro returns an array containing the consecutive values of multiple variables. Users need to define an array to store the return value of this macro

**Usage**

System.getDatas(channel,dataClass,address,count)

**Parameter Description**

- channel (channel number): Integer type, range [1, 16]
- dataClass (data type of the first variable address): Numeric type, with specific meanings as follows

=0 User；

=1 Mcm；

=2 Sys；

=3 Reg；

=4 Bus；

=5 Com；

- address (address of the first variable): Numeric type
- count (number of consecutive variables): Numeric type

**Example**

Example 1: Retrieve the values of #REG100 to #REG104.

```
//Define an array with a size of 5.
var array=new Array(5)
//Initialize the values in the array.
for(var i=0;i<5;i++)
```

array[i]=0

//Read consecutive variables and assign their values to an array.

array=System.getDatas(3,0,100,5)

//Utilize the array array.

for(var j=0;j<5;j++)

total +=array[j]

## 7.2.22. SetDatas (Set multiple consecutive variable values)

Used to set values for consecutive variables. Note: When the array size is not consistent with the specified count, only the smaller number will take effect. No return value.

**Usage**

System.setDatas(channel,dataClass,address,count,arrays)

**Parameter Description**

- Parameters channel, dataClass, address, and count are described in Section 4.1.21.1 for details.
- arays: Arrays for setting variable values

**Example**

Example 1: Set the values of #REG100 to #REG104 to the values in the array. // Define an array with a size of 5.

var array=new Array(5)

// Initialize the values in the array.

for(var i=0;i<5;i++)

array[i]=123.456

// Set multiple consecutive variables, with each variable value corresponding to the numerical values in the array

System.setDatas(3,0,100,5,array)

## 7.2.23. GetBits (Get values of multiple consecutive bits)

This macro is used to obtain the values of consecutive multiple bits. It returns the values of consecutive multiple bits, where each bit value is of boolean type (true/false).(true/false)

**Usage**

System.getBits(channel, bitType, address, bitIndex, count)

**Parameter Description**

- channel (Channel Number): Numeric type, range [1, 15].
- bitType (Type of the first bit): Numeric type, with the following specific meanings:

=0 I_BIT；

=1 O_BIT；

=2 C_BIT；

=3 S_BIT；

=4 A_BIT；

=5 Cn_BIT；

=6 Tm_BIT；

=7 User_BIT；

=8 Mcm_BIT；

=9 Sys_BIT；

=10 Reg_BIT；

=11 Bus_BIT；

=12 Com_BIT；

- address (Starting address): Numeric type

- bbitIndex (Starting bit index): Numeric type. If the data bit type is User_Bit, Mcm_Bit, Sys_Bit, Reg_Bit, Com_BIT, this specifies the initial bit representing a specific bit of data, ranging from [0, 31].
- count (Number of consecutive bits): Numeric type

**Example**

Example 1: Retrieve the values of BIT0 to BIT4 from #USR100.

```
// Define an array with a size of 5.
var array=new Array(5)
// Initialize the values in the array.
for(var i=0;i<5;i++)
array[i]=0
// Read values of consecutive bits and assign them to an array.
array=System.getBits(1, 7, 100,0,5)
// Use the array named 'array'.
if(array[0]==1)
total +=1
```

## 7.2.24. SetBits (Set values of multiple consecutive bits)

To set values for consecutive multiple bits. Note: When the array size differs from the specified count, only the smaller number takes effect. No return value.

**Usage**

```
System.setBits(channel, bitType, address, bitIndex, count,arrays)
```

**Parameter Description**

- Parameters channel, bitType, address, bitIndex, and count are explained in Section 4.1.23.1. Please refer to that section for details.
- arrays (Arrays): Set bit states where each bit state is either 0 or 1.

**Example**

Example 1: Set the values of BIT0 to BIT4 in #USR100 to the values from the array.

```
// Define an array with a size of 5.
var array=new Array(5)
// Initialize the values in the array.
for(var i=0;i<5;i++)
array[i]=1
// Set values for consecutive multiple bits where each bit's state corresponds to the status in the array.
array=System.setBits(1,7,100,0,5,array)
```

## 7.2.25. GetUsbDevicePath (Get USB drive path)

Used to retrieve the directory of a USB drive, returning a value of string type.

**Usage**

```
System.getUsbDevicePath()
```

**Parameter Description**：None

**Example**

Example 1: Retrieve the file directory of the USB drive.

```
var dir = System.getUsbDevicePath()
```

## 7.2.26. GetPrePrgFileSub (Get pre-parsed subprogram file name)

Used to retrieve the filename of the (pre-fetch) subroutine.

**Usage**

System.getPrePrgFileSub()

**Parameter Description**：channel

**Example**

Example 1: Filename of the（pre-fetch )subroutine for Channel 1.

var dir = System.getPrePrgFileSub(1)

## 7.2.27. GetPrePrgFileMain (Get pre-parsed main program file name)

Used to retrieve the filename of the（pre-fetch ) main program.

**Usage**

System.getPrePrgFileMain()

**Parameter Description**：channel

**Example**

Example 1: Filename of the（Pre-fetch） main program for Channel 1.

var dir = System.getPrePrgFileMain(1)

## 7.2.28. SetCurScreenSize (Set current screen size)

Used to set the current screen size.

**Usage**

System.setCurScreenSize()

**Parameter Description**：

(int)width

(int)height

**Example**

Example 1: Set the current screen size to 800*600.

System.setCurScreenSize(800,600)

## 7.2.29. HPrint (Print information on the terminal)

Used for printing information to the terminal.

**Usage**

System.hPrint()

**Parameter Description：**(string)string - The string to be printed.

**Example**

Example 1: Filename of the main program for pre-fetch.

System.hPrint("someText")

## 7.2.30. KeyToString (Convert key value to key text)

Conversion of key codes to key text

**Usage**

System.keyToString()

**Parameter Description：**

(int)value:Key value

Key code reference:"https://doc.qt.io/qt-5/qt.html#Key-enum"

**Example**

Example 1: var keyStr = System.keyToString(0x42)

print(keyStr)//Letter B

## 7.2.31. StringToKey (Convert key text to key value)

Conversion of key text to key code

**Usage**

System.stringToKey()

**Parameter Description：**

(string)value:Key text

**Example**

Example 1: var keyStr = System.stringToKey("B")

print(keyStr)//0x42

## 7.2.32. ConvertCompressCnc (Convert file to CNC format)

Conversion of files to CNC format

**Usage**

System.convertCompressCnc()

**Parameter Description：**

(string)cncFileName:File name, including full path

**Example**

Example 1: var result =

System.convertCompressCnc("/usr/sys0001/program/program.CNC")

Printing the result in CNC format

## 7.2.33. SimulateHandle (Simulate touchscreen operation)

Simulating touchscreen operations

**Usage**

System.simulateHandle()

**Parameter Description：**

(int)x: x coordinate

(int)y: y coordinate

(double)doubleClick: Whether it is a double-click

=true Double-click

=false Single-click

**Example**

Example 1:System.simulateHandle(400,300,true)

## 7.2.34. SetInputMethod (Set input method for switching between Chinese and English, mini soft keyboard)

Setting up switching between Chinese and English input for a mini virtual keyboard

**Usage**

System.setInputMethod()

**Parameter Description**

(int)method:Chinese and English

=0 Switch to English input

=1 Switch to Simplified Chinese input

**Example**

Example 1:System.setInputMethod(1)

## 7.3. HMI Function Group

## 7.4. Math Function Group

## 7.5. String Function Group

## 7.6. Number Function Group

## 7.7. HInputDialog Function Group

## 7.8. HMessageBox Function Group

## 7.9. Hfile Function Group

## 7.10. HFileEdit Function Group

## 7.11. HFileInfo Function Group

## 7.12. HFileVar Function Group

## 7.13. HAuthorityManagement Function Group

## 7.14. HErrorDialog Function Group

## 7.15. HProperty Function Group

## 7.16. HSharedFile Function Group

## 7.17. HCamera Function Group

## 7.18. HVisionModule Function Group

# 8. Others

FINGER CNC


**Guangzhou Finger Technology Co.,Ltd**

Hotline：020-39389901
Repair Helpline：18127931302
Fax：020-39389903
Postal Code：511495
E-mail：finger@fingercnc.com
Website：www.finger-cnc.com
Address：201,No. 8, Chengding Street, Zhongcun Street, Panyu District, Guangzhou City, Guangdong Province


Official Website

Official Wechat